第 3 章　关系数据库的基本理论

通过第 2 章的学习，读者对数据库系统已经有了全面的认识。目前应用最广泛的是关系数据库，本章将从关系模型概述、关系数据结构、关系的完整性约束、关系操作集合和关系规范化等方面介绍关系数据库的基本理论。为了使读者更容易理解关系数据库理论知识，在每个知识点辅以详细案例，增加读者的感性认识。

3.1　关系模型概述

关系数据库以关系模型为基础，关系模型是关系数据库的基础，它由数据结构、关系操作集合和完整性约束这 3 个部分组成。

(1) 数据结构。数据模型所操作的对象、类型的集合。关系数据库中全部数据及其相互联系都被组织成关系(即二维表格)的形式。

(2) 关系操作集合。即数据操作，表现为关系代数的各种运算，是数据模型所允许的操作方式。关系模型提供一组完备的高级关系运算，以支持对数据库的各种操作。

(3) 完整性约束。保证数据有效、正确的约束条件。

3.1.1　关系的定义

关系理论是建立在集合代数的理论基础上，因此可以用集合代数来定义用二维表表示的关系。先从二维表出发理解下面几个基本术语及概念。

1. 关系的逻辑概念

(1) 关系。由行和列组成的二维表。

表示为：关系名(属性 1，属性 2，…，属性 n)或表名(字段 1，字段 2，…，字段 n)。

(2) 元组、属性、度。二维表中的每一行代表一个元组，表的每一列代表一个属性(字段)。属性的个数就是度。

(3) 关键字，简称键。在一个关系模式中，必然存在一个属性(属性组)，当这个属性(属性组)的值确定之后，关系中别的属性的值也就唯一地确定了。这个属性(属性组)就是该关系模式的关键字。或者说，对于某个关系，某个或某几个属性，可以唯一地确定一条记录，他们就是该关系的关键字，关键字可以进行如下分类：

① 超键(super key)。在关系中能唯一标识元组的属性集称为关系模式的超键。如表 3-1 所示的学生关系，假设不存在同名的情况，(姓名，年龄)的组合就是一个超键。而该超键中“年龄”是多余的属性，因为去掉“年龄”后，剩下的属性(姓名)仍能唯一地标识元组。

② 候选键(candidate key)。不含有多余属性的超键称为候选键。对于某个关系，若存在多个属性组都是关键字，则称它们中的每一个都为该关系的候选关键字。显然，候选键必为超键，但超键却不一定是候选键。如表 3-1，该关系就有“学号”和“姓名”两个候选键。而注意区别的是，超键强调的是“唯一标识”；候选键则在“唯一标识”的基础上还要求不含多余

属性的“最小属性组合”。

③ 主键(primary key)。从候选键中任选一个作为现行关键字,则该关键字称为主键。一个关系的主键只有一个,主键一经选定通常不变,如表 3-1 的关系中,可以从候选键“学号”或者“姓名”中选定一个作为该关系的主键。

④ 外键(foreign key)。不是本关系的关键字,而是另一个关系的关键字。关系之间的连接常常依靠外部关键字来实现。

例如,在表 3-1 所示的学生关系中,学号是主键,而在表 3-2 所示选修关系中,学号和选修课程号的组合为选修关系的主键,学号只能作为选修关系的外键,与学生关系表的学号相对应,并参照学生关系表的学号进行取值。

表 3-1 学生关系

学号	姓名	性别	年龄	系别	政治面貌
2000001	刘文华	女	19	计算机	团员
2000002	王明	男	19	计算机	团员
2000003	李刚	男	18	计算机	团员
2000004	吴惠珊	女	19	计算机	团员
2000005	吴文	男	19	艺术	团员
2000006	冯丽英	女	18	艺术	团员

表 3-2 选修关系

学号	选修课程号	成绩	学号	选修课程号	成绩
2000001	03	66	2000004	08	67
2000002	05	88	2000005	02	78
2000003	05	99	2000006	07	58

2. 关系的数学概念

在初步理解上面的基本术语及概念后,现在从数学的概念了解关系的定义。

(1) 域(domain)。域是一组具有相同数据类型的值的集合。如:{0,1},{中国,美国},{语文,数学,英语}等都可以是域。每个域要有域名,域中数据的个数称为域的基数(即集合内元素的个数)。例如:

D_1={0,1},表示数字的集合。

D_2={中国,美国},表示国家的集合。

D_3={语文,数学,英语},表示课程的集合。

这里的 D_1,D_2,D_3 是域名,D_1 的基数为 2,D_2 的基数为 2,D_3 的基数为 3。由于计算机及其存储系统不能实现无限的集合,故在数据库中,域总是包含着有穷多个值的有限集。

(2) 笛卡儿积(Cartesian product)。

定义:给定一组域 $D_1,D_2,\cdots,D_n$,定义 $D_1,D_2,\cdots,D_n$ 的笛卡儿积为

$$D_1 \times D_2 \times \cdots \times D_n = \{(d_1,d_2,\cdots,d_n) \mid d_i \in D_i, i = 1,2,\cdots,n\}$$

其中,每一个元素 $(d_1,d_2,\cdots,d_n)$ 叫做一个 n 元组(n-tuple),或简称为元组(tuple)。即当

$n=1$ 时，称一元组或单元组；$n=2$ 时，称二元组；…；$n=p$ 时，称 p 元组。

元素中的每一个值 d_i 叫做一个分量(component)。

若 $D_i(i=1,2,\cdots,n)$为有限集，其基数(Cardinal number)为 $m_i(i=1,2,\cdots,n)$，则 $D_1\times D_2\times\cdots\times D_n$ 的基数为

$$M=\prod_{i=1}^{n}m_i$$

式中，M 为笛卡儿积的基数；

m_i 为第 i 个域的基数；

n 为域的个数。

例如，有 3 个域 D_1,D_2,D_3，各域的基数分别为 $m_1=2$，$m_2=3$，$m_3=6$，那么笛卡儿积 $D_1\times D_2\times D_3$ 的基数为 $2\times3\times6=36$。也就是说，该笛卡儿积有 36 个元组，每个元组有 3 个分量(三元组)。

(3) 关系(relation)。设有属性 $A_1,A_2,\cdots,A_n$，它们分别在值域 $D_1,D_2,\cdots,D_n$ 中取值，这些值域中的任意一个子集称为一个关系，记为

$$R(A_1,A_2,\cdots,A_n)$$

$$R\in D_1\times D_2\times\cdots\times D_n$$

其中，R 为关系名；n 为关系 R 的度数。一个 n 度关系就有 n 个属性。

3.1.2 关系的性质

通过前面的介绍知道，一个关系就是一张二维表，但并不是所有的二维表都是关系。关系是一种规范化的二维表，作为关系的二维表必须满足下列性质。

(1) 关系中的每一个属性必须是不可再分的。

该属性说明关系中所有属性都是原子的，即是一个确定的属性，而不是属性的集合，即不可“表中有表”。属性值可以为空值，表示“未知”或“不可使用”。满足该性质的关系被称为规范化关系，否则为非规范化关系。

例如，在表 3-3 中，课程名含有微积分、大学英语两项，出现了“表中有表”的现象，则为非规范化关系，因而把课程名分成微积分、大学英语两列，将其规范化，如表 3-4 所示。

表 3-3 非规范关系

姓 名	课程名	
	微积分	大学英语
刘文华	85	92
王明	90	90

表 3-4 规范关系

姓 名	微积分	大学英语
刘文华	85	92
王明	90	90

(2) 每一列中的数值是同类型的数据，来自同一个域。

(3) 不同的列应给予不同的属性名。不同的属性可来自同一个域，即它们的分量可以取自同一个域。

例如，有表示职工工作的域{辅导员，教师，工人，公务员}，在表 3-5 所示关系中，职业与兼职是两个不同的属性，但它们都可以在职工工作的域中取值。

表 3-5　职工信息关系

姓　名	职　业	兼　职
莫宏	教师	辅导员
徐晓丽	工人	教师
刘均	教师	辅导员

(4) 同一关系中不允许有相同的元组,即任意两个元组不能完全相同。

(5) 行、列的次序可以任意交换,不影响关系的实际意义。如表 3-6～表 3-8 表示为同一个关系。

表 3-6　学生关系(a)

姓名	性别
李平	女
张军	男

表 3-7　学生关系(b)

姓名	性别
张军	男
李平	女

表 3-8　学生关系(c)

性别	姓名
女	李平
男	张军

(6) 关系是随着时间的推移而不断变化的。关系会改变,这是由于对数据库的操作会引起元组的插入、删除或更新的缘故。

3.1.3　关系模式

在数据库中要区分型和值的概念。关系数据库中,关系模式是型,关系是值。关系模式是对关系的描述,而在关系模型中最主要的组成成分是关系。一个关系就是一张二维表,表中一行称为一个元组(tuple),表中一列称为一个属性(attribute),每一列对应一个唯一的名字称为属性名,属性的取值范围称为属性的域。关系是元组的集合,一个元组由属性值组成。关系有 M 列,则称该关系是 M 元关系。关系模式常记为:

$$\text{Rel-Name}(A_1, A_2, A_3, \ldots, A_n)$$

其中,Rel-Name 是关系名,$A_i(1 \leqslant i \leqslant n)$是属性名。

例如:

学生(学号,姓名,性别,年龄,系别);

教师(教师号,姓名,性别,年龄,系别);

课程(课程号,课程名,课时);

选课(学号,课程号,成绩);

授课(教师号,课程号)。

分别表示“学生”关系模式、“教师”关系模式、“课程”关系模式、“选课”关系模式和“授课”关系模式。

关系实际上就是关系模式在某一时刻的状态或内容(如表 3-9～表 3-11 所示)。也就是说,关系模式是型,关系是它的值。关系模式是静态的、稳定的,而关系是动态的、随时间不断变化的,因为关系操作在不断地更新着数据库中的数据。但在实际应用当中,常常把关系模式和关系统都称为关系。

表 3-9 “教师”关系

教师号	姓名	性别	年龄	职称	系别
T01	王伟	男	47	教授	计算机
T02	张兰	女	28	讲师	信息
T03	陈强	男	30	讲师	计算机
T04	周敏	女	52	教授	自动化
T05	叶善	女	39	副教授	信息

表 3-10 “课程”关系

课程号	课程名	课时
C01	程序设计	60
C02	操作系统	33
C03	数字逻辑	55
C04	数据结构	80
C05	数据库	60

表 3-11 “授课”关系

教师号	课程号
T01	C04
T02	C05
T03	C01
T04	C03
T05	C02

3.2 关系数据结构

3.1 节简单地介绍了关系模型的特点和表现形式，关系模型的数据结构能够描述出现实世界的实体以及实体间的各种联系。关系数据结构是关系模型的一部分，是关系模型的进一步细化和描述。关系模型数据结构可以由 E-R 图导出，并且存在着自己的体系结构。

3.2.1 从 E-R 图导出关系模型数据结构

E-R 图纯粹是描述信息的结构，是构造数据模型的依据，属于概念模型。数据库的逻辑设计需要把概念模型转换为具体 DBMS 能处理的数据模型，不同的数据模型，其转换规则不同。相互关联的关系模式的集合构成一个关系模型，从 E-R 模型向关系模型转换时，所有实体和联系都要转换成相应的关系模式。

从 E-R 图出发导出关系模型数据结构有两点原则。

第一，对 E-R 图中的每个“实体集”，都应转换成一个关系，该关系内至少要包含对应实体的主要属性，并应根据关系所表达的语义确定哪个属性或哪几个属性组作为“主键”，主键用来标识实体。

第二，对 E-R 图中的“联系”，要根据实体联系的方式，采取不同的方法加以处理，有时需把“联系”自身用一个关系来表示，有时则无须专门用一个关系表示“联系”，而是把“联系”纳入表示实体的关系中。不同联系方式的 E-R 图转换成关系数据结构时，应该遵循下面 3 个规则。

(1) 两实体集间 1∶n 联系。两实体集间 1∶n 联系，可将“1 方”实体的主关键字纳入“n 方”实体集对应的关系中作为“外部关键字”，同时把联系的属性也一并纳入“n 方”对应的关系中。

【例 3-1】 导出图 3-1(b)所示学校与教师联系的关系数据结构。

从图 3-1(a)所示的 E-R 图中，存在两个实体，实体的关系模式如下：

学校(学校名，校址，校长)；

教师(教工号，姓名，专长)。

因为该 E-R 图为 1∶n 关系，所以应将联系“聘任”的属性“年薪”并入到“n 端”教师关系中，同时将“1 端”的主键“学校名”也并入到“n 端”教师关系中。创建好的数据结构如图 3-1(b)所示。

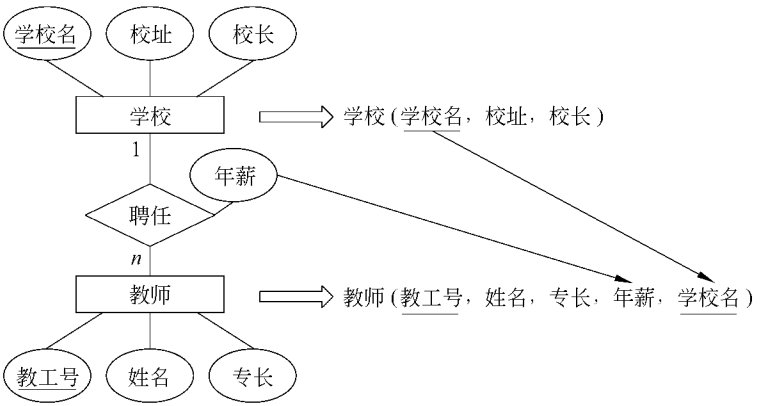


图 3-1 学校与教师关系的 E-R 图和关系数据结构

(2) 两实体集间 m∶n 联系。对于两实体集间 m∶n 联系，必须对“联系”单独建立一个关系，而且联系关系的属性至少要包括被它所联系的双方实体集的“主键”，如果联系有属性，也要归入这个关系中。

【例 3-2】 导出图 3-2(b)所示学生与课程联系的关系数据结构。

从图 3-2(a)所示的 E-R 图中，存在两个实体，实体的关系模式如下：

学生(学号，姓名，性别)；

课程(课程号，课程名，学分)。

因为该 E-R 图为 m∶n 关系，所以应将联系“选修”单独作为一个关系，而“学生”的主键“学号”并入到“选修”关系中作为外键，同时将而“课程”的主键“课程号”也并入到“选修”关系中。创建好的数据结构如图 3-1(b)所示。

(3) 两实体集间 1∶1 联系。假设 A 实体集与 B 实体集是 1∶1 的联系，联系的转换有 3 种方法。

方法 1：把 A 实体集的主关键字加入到 B 实体集对应的关系中，如果联系有属性也一并加入。

方法 2：把 B 实体集的主关键字加入到 A 实体集对应的关系中，如果联系有属性也一并加入。

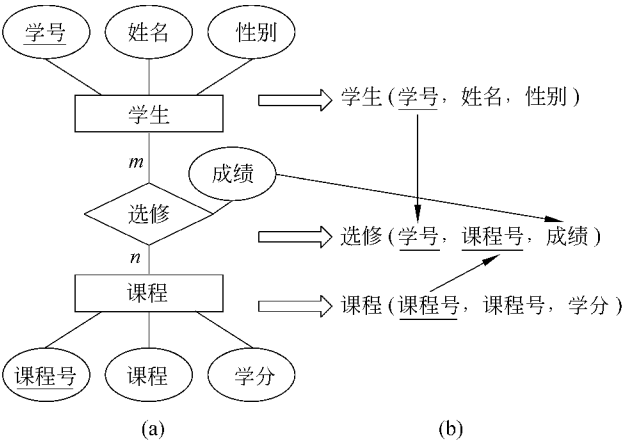


图 3-2　学生与课程关系的 E-R 图和关系数据结构

方法 3：建立第 3 个关系，关系中包含两个实体集的主关键字，如果联系有属性也一并加入。

对于方法 3，由于要为联系建立新的关系，从简便的角度考虑一般不采用，只采用方法 1 和方法 2。

【例 3-3】 导出图 3-3(b)所示校长与学校联系的关系数据结构。

从图 3-3(a)所示的 E-R 图中，存在两个实体，实体的关系模式如下：

校长(姓名，性别，年龄)；

学校(学校名，校址，类别，姓名)。

因为该 E-R 图为 1∶1 关系，所以无须将联系“管理”单独作为一个关系，而只需把“校长”的主键“姓名”并入到“学校”关系中作为外键，或者将“学校”的主键“学校名”也并入到“校长”关系中作为外键。创建好的数据结构如图 3-3(b)所示。

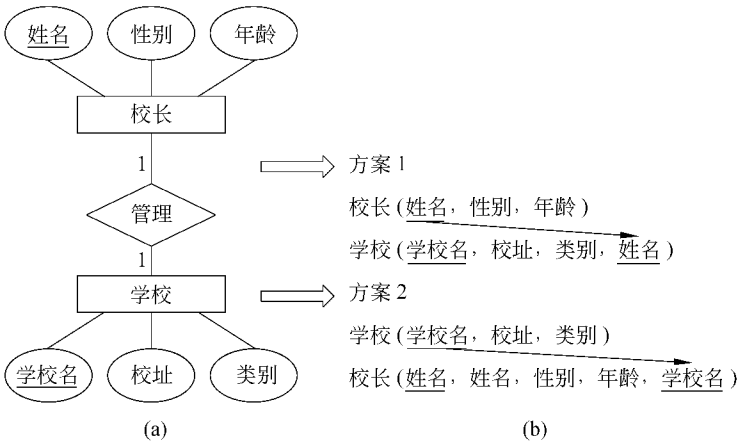


图 3-3　校长与学校关系的 E-R 图和关系数据结构

3.2.2　关系模型的体系结构

关系模型基本上遵循数据库的三级体系结构：关系概念模式、关系内模式和关系外模式。

(1) 关系概念模式。一个关系模式对应一个二维表框架,关系概念模式是由若干个关系模式组成的集合,描述关系数据库中全部数据的整体逻辑结构。

例如学生信息数据库中有学生关系模式、课程关系模式和选修关系模式 3 个关系模式:

学生(学号,姓名,性别);

课程(课程号,课程名,学分);

选修(学号,课程号,成绩)。

这 3 个关系模式集就构成学生信息数据库的概念模式。

(2) 关系外模式。是关系概念模式的一个逻辑子集,描述关系数据库中数据的局部逻辑结构。通过关系运算可从一个关系概念模式中导出多个关系外模式。

例如从学生信息数据库的概念模式导出学生选课信息模式:

学生选课(学号,姓名,课程名,成绩)

该模式就是学生信息数据库的外模式。

(3) 关系内模式。关系存储时的基本组织方式是文件,一个关系对应一个文件。总的来说,关系数据库的内模式是一组数据文件(包括索引等)的集合。

例如,在第 1 章案例中创建的 Student 表、Course 表和 SC 表就是数据库 StudentInfo 的内模式。

3.3 关系的完整性约束

关系模型有 3 类完整性约束条件:实体完整性约束、参照完整性约束和用户定义完整性约束。这 3 类约束条件中前两类是关系模型必须满足的完整性约束条件,由关系系统自动支持,而后一类约束条件是用户针对特定的数据库设置的约束条件。

3.3.1 实体完整性

实体完整性规则指的是关系中的所有元组,在组成主键的属性上不允许取空值。

这个规则很容易理解,因为主属性能唯一标识关系中的元组,若取空值,便失去唯一元组功能。例如关系模式:学生(学号,姓名,性别,年龄,籍贯,专业名称),其中“学号”是主键,而主键对应的属性只有“学号”,所以“学号”也是主属性。根据实体完整性约束规则,“学号”不能取空值。在学生选课关系模式中,选修(学号,课程编码,成绩)中,属性组“学号”和“课程编码”为主键,同时也是主属性,所以这两个属性均不能取空值。

实体完整性规则是针对基本关系而言,即针对现实世界的一个实体集,而现实世界中的实体是可区分的。该规则的目的是利用关系模式中的主键或主属性来区分现实世界中的实体集中的实体,所以不能取空值。

3.3.2 参照完整性

参照完整性规则指的是不能引用不存在的实体。

在关系模型中,实体与实体之间的关联是采用关系模式来描述的。通过引用对应实体的关系模式的主键来表示对应实体之间的关联。

例如,关系模式:部门(部门编码,部门名称,电话,办公地址),职工(职工编码,姓名,性

别，年龄，籍贯，所属部门编码）。

其中，职工关系模式中的“所属部门编码”与部门关系模式中的主键“部门编码”相对应，所以“所属部门编码”是职工关系模式的外键。职工关系模式通过外键来描述与部门关系模式的关联。职工关系中的每个元组（每个元组描述一个职工实体）通过外键表示该职工所属的部门。当然，被参照关系的主键和参照关系的外键可以同名，也可以不同名。被参照关系与参照关系可以是不同关系，也可以是同一关系。

参照完整性规则简单地说，就是从 A 关系的外键出发去找 B 关系的记录，必须能够找到。

例如在职工关系中，某一个职工“所属部门编码”要么取空值，表示该职工未被分配到指定部门。要么等于部门关系中某个元组的“部门编码”，表示该职工隶属于指定部门。若既不为空值，又不等于被参照关系——部门中某个元组的“部门编码”分量，表示该职工被分配到一个不存在的部门，则违背参照完整性规则。所以，参照完整性规则就是定义外键与主键之间的引用规则，也是关系模式之间关联的规则。

3.3.3 用户定义完整性

用户定义完整性是针对某一具体数据库的约束条件，它反映某一具体应用所涉及的数据必须满足的语义的要求，关系模型应提供定义和检验这一类完整性的机制，以便用统一的系统的方法处理它们，而不是由应用程序来承担这一功能。

例如，在职工关系中，职工年龄分量的取值范围应该限定在 18～60 之间，学生选课的成绩取值范围应该限定在 0～100 之间，关系模型应该为用户提供定义和检验这一类完整性约束机制，保证数据的正确性。

3.4 关系操作集合

关系操作集合就是对关系中数据的操作运算，主要表现为关系代数。关系代数是一种抽象的查询语言，在该语言中，将关系作为运算的对象，用关系运算的结果来表达查询。

关系代数的运算对象是关系，运算结果也为关系。关系代数包含 4 类运算：集合运算、专门的关系运算、算术比较运算和逻辑运算，各类运算使用的运算符及含义如表 3-12 所示。

比较运算符和逻辑运算符是用来辅助专门的关系运算符进行操作的，所以关系代数的运算通常分为传统的集合运算和专门的关系运算两类。

3.4.1 传统的集合运算

传统的集合运算是二目运算，是在两个关系中进行的。但是并不是任意的两个关系都能进行这种集合运算，而是要在两个满足一定条件的关系中进行运算。

给定两个关系 R、S，若 R 和 S 满足：

(1) 具有相同的度 n；

(2) R 和 S 两个关系要相容。

即

两个关系具有相同的关系模式；

表 3-12　关系代数运算符

运算符		含义	运算符		含义
集合运算符	∪	并	算术比较运算符	>	大于
	−	差		⩾	大于等于
	∩	交		<	小于
	×	广义笛卡儿积		⩽	小于等于
专门的关系运算符	σ	选择		=	等于
	Π	投影		≠	不等于
	⋈	连接	逻辑运算符	¬	非
	÷	除		∧	与
				∨	或

两个关系对应的属性列都出自同一个域。

除笛卡儿积外，要求参加运算的关系必须满足上述的相容性定义。

满足上述条件的两个关系才可以进行传统的集合运算。集合运算包括并、交、差、广义笛卡儿积 4 种运算。

1. 并(union)运算

设关系 R 和关系 S 具有相同的度 n(即两个关系都有 n 个属性)，且相应的属性取自同一个域，则关系 R 与关系 S 的并由属于 R 或属于 S 的所有元组组成。其结果关系仍为 n 度关系。记作：

$$R \cup S = \{t \mid t \in R \vee t \in S\}$$

该运算的示意图如图 3-4(a)所示。

2. 差(difference)运算

设关系 R 和关系 S 具有相同的目 n，且相应的属性取自同一个域，则关系 R 与关系 S 的差由属于 R 而不属于 S 的所有元组组成。其结果关系仍为 n 目关系。记作：

$$R - S = \{t \mid t \in R \wedge t \notin S\}$$

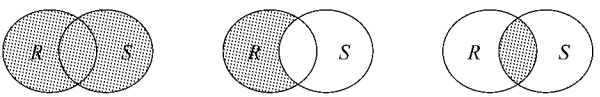
该运算的示意图如图 3-4(b)所示。

3. 交(intersection)运算

设关系 R 和关系 S 具有相同的目 n，且相应的属性取自同一个域，则关系 R 与关系 S 的交由既属于 R 又属于 S 的所有元组组成。其结果关系仍为 n 目关系。记作：

$$R \cap S = \{t \mid t \in R \wedge t \in S\}$$

该运算的示意图如图 3-4(c)所示。



(a) 并 $(P_1 \cup P_2)$　(b) 差 $(P_1 - P_2)$　(c) 交 $(P_1 \cap P_2)$

图 3-4　图形表示并、差及交运算