

关系数据库是创建在关系模型基础上的数据库,借助于集合代数等数学概念和方法来处理数据库中的数据。现实世界中的各种实体以及实体之间的各种联系均用关系模型表示。最早提出“关系模型”的是美国 IBM 公司的 Codd,他发表在美国计算机学会会刊上题为《大型共享数据库的关系模型》的论文首次提出了数据库的关系模型。由于关系模型简单明了,具有坚实的数学理论基础,所以一经推出就受到了学术界和产业界的高度重视和广泛响应,并很快成为数据库市场的主流。数据库领域当前的研究工作大都以关系模型为基础。本章主要介绍关系数据库中的关系模型、关系形式定义、关系的完整性、关系运算的理论以及关系数据库的特点,通过学习使读者了解如何在关系模型基础上建立数据库,如何借助集合代数等数学概念和方法来处理数据库中的数据。

本章导读

- 关系数据库与关系模型
- 关系的形式化定义
- 关系完整性
- 关系运算

3.1 关系数据库与关系模型

关系数据库是建立在关系数据库模型基础上的数据库,借助于集合代数等概念和方法来处理数据库中的数据。关系数据库在关系数据库管理系统(RDBMS)的帮助下得以发展,我们今天所使用的绝大部分数据库系统都是关系数据库,包括 Oracle、SQL Server、MySQL、Sybase、DB2 等。关系数据库使用 SQL 进行查询,结果集通过访问一个或多个表的查询生成。

关系模型有严格的数学基础,抽象级别比较高,而且简单清晰,便于理解和使用。关系数据模型提供了关系操作的特点和功能要求,但不给 DBMS 的语言给出具体的语法要求。对关系数据库的操作是高度非过程化的,用户不需要指出特殊的存取路径,路径的选择由 DBMS 的优化机制来完成。

关系数据模型是以集合论中的关系概念为基础发展起来的。关系模型中无论是实体还是实体间的联系均由单一的结构类型——关系来表示。在实际的关系数据库中的关系也称为“表”,一个关系数据库就是由若干个表组成的。

关系模型是用二维表的形式表示实体和实体间联系的数据模型,关系模型由关系数据结构、关系操作、关系完整性约束三部分组成。

1. 关系数据结构

关系模型的数据结构比较单一。在关系模型中,现实世界的实体与实体间的联系均用关系来表示。一般用二维表的形式表示实体和实体间联系。从用户的角度来看,关系模型中数据的逻辑结构就是一张二维表,如表 1.3.1 和表 1.3.2 所示。

表 1.3.1 学生关系表

学 号	姓 名	性 别	年 龄
20190101101	陈名军	女	20
20190101102	吴小晴	女	23
20190101103	李国庆	男	21
20190101104	李祥	男	21
20190101105	王成	男	22

表 1.3.2 选课关系表

学 号	课 程 号	成 绩
20190101101	101	60
20190101101	102	83
20190101102	201	78
20190101102	202	87

二维表一般具有下面几个性质。

(1) 根据使用的 DBMS 不同,表中元组的个数也不同。

(2) 表中不能存在完全相同的元组,即二维表中有相应的表级约束,元组应各不相同。

(3) 表中元组的次序无要求,即二维表中元组的次序可以任意交换。

(4) 表中分量必须取原子值,即二维表中每一个分量都是不可分割的数据项。如表 1.3.1 中的列“性别”不能再分为两个或两个以上数据项。

在表 1.3.3 中,“学期”出现了“表中有表”的现象,是非规范化关系。将学期分为“第一学期”和“第二学期”,即可使其规范化,如表 1.3.4 所示。

表 1.3.3 非规范开课情况表

课 号	课 程 名 称	学 期	
		第一学期	第二学期
001	数据库原理	40 学时	60 学时
002	体育	40 学时	40 学时
003	英语	80 学时	80 学时

表 1.3.4 规范开课情况表

课 号	课 程 名 称	第 一 学 期	第 二 学 期
001	数据库原理	40 学时	60 学时
002	体育	40 学时	40 学时
003	英语	80 学时	80 学时

(5) 表中属性的顺序无要求,即二维表中的属性与顺序无关,可任意交换。如“学号”所在列可以和任意列交换位置,不影响查询、删除、更新、插入等操作。

(6) 表中分量值域有同一性,即二维表中的属性分量属于同一值域。如表中的“学号”都是字符型,宽度为 10。

(7) 表中不同的属性要给予不同的属性名,即表中的每一列为一个属性,但不同的属性可出自同一个域,如表 1.3.5 所示的关系,“教材”与“参考书”是两个不同的属性,但它们取自同一个域。

书 = {数据库原理实用教程, 计算机数学, 高等数学...}

表 1.3.5 课程表

课 号	课 程 名 称	教 材	参 考 书
001	数据库原理	数据库原理实用教程	数据库应用
002	数学	高等数学	计算机数学
003	英语	大学英语	实用英语

(8) 关系模型要求关系必须是规范化的,即要求关系必须满足一定的规范条件。

2. 关系操作

关系操作采用集合的操作方式。关系模型中常用的操作包括选择(Select)、投影(Project)、连接(Join)、除(Divide)、并(Union)、交(Intersection)、差(Except)等查询操作。

3. 关系的三类完整性约束

关系模型中允许定义三类完整性约束: 实体完整性约束、参照完整性约束、用户定义的完整性约束。其中实体完整性是规定表的每一行在表中是唯一的实体。参照完整性是指两个表的主关键字和外关键字的数据应一致,保证了表之间的数据的一致性,防止了数据丢失或无意义的数据在数据库中扩散。这两个约束是由关系系统自动支持的。用户定义的完整性是针对某个特定关系数据库的约束条件,它反映某一具体应用必须满足的语义要求。

3.2 关系的形式化定义

在关系模型中,无论是实体还是实体之间的联系均由单一的结构类型即关系(表)来表示,关系是建立在集合基础之上的,现在从集合的角度给出关系的形式化定义。

3.2.1 关系相关概念

1. 域

域(Domain)是一组具有相同数据类型的值的集合,又称为值域(用 D 表示)。

如自然数、整数、实数,长度小于 10 字节的字符串集合,{1,2}、介于某个取值范围的整数(如 20~100)、介于某个取值范围的日期等,都可以称为域。

(1) 域中所包含的值的个数称为域的基数(用 m 表示)。

(2) 关系中用域表示属性的取值范围,如表 1.3.1 所示的学生关系表。

$$D_1 = \{20190101101, 20190101102, 20190101103, 20190101104, 20190101105\}$$
$$D_2 = \{\text{陈名军, 张小平, 李国庆, 李祥, 王成}\}$$

$$m_1 = 5$$
$$m_2 = 5$$

$$D_3 = \{\text{男}, \text{女}\}$$

$$m_3 = 2$$

$$D_4 = \{20, 23, 21, 22\}$$

$$m_4 = 4$$

其中, D_1 、 D_2 、 D_3 、 D_4 为域名, 分别表示学生关系中学号、姓名、性别、年龄的集合。

2. 元组

关系表中的一行称为一个元组(Tuple)。元组可表示一个实体或实体之间的联系, 是属性的有序多重集。

3. 码

在二维表中, 用来唯一标识一个元组的某个属性或属性组合称为该表的键或码(Key), 也称为关键字, 如表 1.3.1 中的属性“学号”。码必须是唯一的。如果一个二维表中存在多个关键字或码, 它们称为该表的候选关键字或候选码。在候选关键字中指定一个关键字作为用户使用的关键字称为主关键字或主码。二维表中某个属性或属性组合虽不是该表的关键字或只是关键字的一部分, 但却是另一个表的关键字时, 称该属性或属性组合为这个表的外码或外键。下面具体介绍码的相关概念。

码是数据库系统中的基本概念, 是能唯一标识实体的属性, 它是整个实体集的性质, 而不是单个实体的性质。码包括超码、候选码、主码和外码。

(1) 超码(Super Key)是一个或多个属性的集合, 这些属性可以让我们在一个实体集中唯一地标识一个实体。一个关系可能有多个超码。如果 K 是一个超码, 那么 K 的任意超集也是超码, 也就是说如果 K 是超码, 那么所有包含 K 的集合也是超码。

(2) 候选码(Candidate Key)是从超码中选出的, 自然地, 候选码也是一个或多个属性的集合。一个关系可能有多个候选码。候选码是最小超码, 它们的任意真子集都不能成为超码。例如, 如果 K 是超码, 那么所有包含 K 的集合都不能是候选码; 如果 K 、 J 都不是超码, 那么 K 和 J 组成的集合 (K, J) 有可能是候选码。

(3) 主码(Primary Key), 是从多个候选码中任意选出的一个, 如果候选码只有一个, 那么该候选码就是主码。虽然说主码的选择是比较随意的, 但在实际开发中还是要靠一定的经验, 不然开发出来的系统会出现很多问题。一般来说主码都应该选择那些从不或者极少变化的属性。

(4) 外码(Foreign Key), 在关系 K 中的属性或属性组若在另一个关系 J 中作为主码使用, 则称该属性或属性组为 K 的外码。 K 的外码和 J 中的主码必须定义在相同的域上, 允许使用不同的属性名。

4. 属性

关系中不同列可以对应相同的域, 为了加以区分, 必须给每列起一个名字, 称为属性(Attribute)。

5. 分量

分量(Component)是元组中的一个属性的值, 如 20190101101, 陈名军。

6. 笛卡儿积

设有一组域 $D_1, D_2, \dots, D_n$, 这些域可以部分或者全部相同, 也可以完全不同。 $D_1, D_2, \dots, D_n$ 的笛卡儿积(Cartesian Products)为:

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

其中:

元组——每一个元素 $(d_1, d_2, \dots, d_n)$ 称为一个 n 元组(n -Tuple)或简称元组(Tuple)。
分量——元组中每一个 d_i 值称为一个分量。

基数(Cardinal Number): 若 $D_i (i=1, 2, \dots, n)$ 为有限集,其基数为 $m_i (i=1, 2, 3, \dots, n)$, 则 $D_1 \times D_2 \times D_3 \times \dots \times D_n$ 的基数为:

$$M = \prod_{i=1}^n m_i \quad (i=1, 2, 3, \dots, n)$$

设 $D_1 = \{A, B, C\}, D_2 = \{1, 2\}$, 则 $D_1 \times D_2 = \{(A, 1), (A, 2), (B, 1), (B, 2), (C, 1), (C, 2)\}$ 的基数为: $3 \times 2 = 6$, 如图 1.3.1 所示。

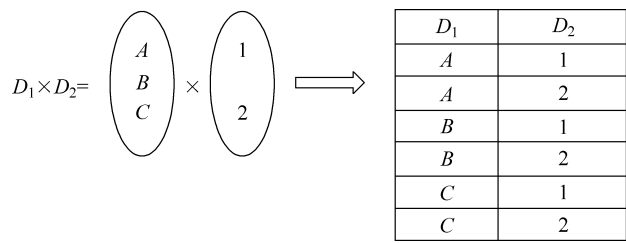


图 1.3.1 笛卡儿积

笛卡儿积可表示为一张二维表,表中的每行对应一个元组,表中的每列对应一个域,但是若干个域的笛卡儿积可能存在大量的数据冗余,因此一般只取其中的某些子集。笛卡儿积的子集就称为关系。

例 3.1 给出三个域:

D_1 = 家电集合 = (冰箱, 电视)

D_2 = 产地集合 = (上海, 深圳)

D_3 = 单价集合 (2000, 8000, 10 000)

则 D_1, D_2, D_3 的笛卡儿积为:

$D_1 \times D_2 \times D_3 = \{(\text{冰箱}, \text{上海}, 2000), (\text{冰箱}, \text{上海}, 8000), (\text{冰箱}, \text{上海}, 10\ 000), (\text{冰箱}, \text{深圳}, 2000), (\text{冰箱}, \text{深圳}, 8000), (\text{冰箱}, \text{深圳}, 10\ 000), (\text{电视}, \text{上海}, 2000), (\text{电视}, \text{上海}, 8000), (\text{电视}, \text{上海}, 10\ 000), (\text{电视}, \text{深圳}, 2000), (\text{电视}, \text{深圳}, 8000), (\text{电视}, \text{深圳}, 10\ 000)\}$
其中, (冰箱, 上海, 2000)、(冰箱, 上海, 8000)等都是元组; 冰箱、上海、2000 等都是分量。

该笛卡儿积的基数为 $2 \times 2 \times 3 = 12$ 。 $D_1 \times D_2 \times D_3$ 共有 12 个元组,这 12 个元组可列成一张二维表,如表 1.3.6 所示。

表 1.3.6 D_1, D_2, D_3 的笛卡儿积

产 品 名 称	产 地	单 价
冰箱	上海	2000
冰箱	上海	8000
冰箱	上海	10 000
冰箱	深圳	2000
冰箱	深圳	8000
冰箱	深圳	10 000
电视	上海	2000

续表

产 品 名 称	产 地	单 价
电视	上海	8000
电视	上海	10 000
电视	深圳	2000
电视	深圳	8000
电视	深圳	10 000

7. 关系

$D_1 \times D_2 \times \cdots D_n$ 的一个子集 R 称为在域 $D_1 \times D_2 \times \cdots D_n$ 上的一个关系 (Relation), 通常将其表示为 $R(D_1 \times D_2 \times \cdots D_n)$, 其中, R 表示该关系的名称, n 称为关系 R 的元数或度数 (Degree), 而关系 R 中所含有的元组数称为 R 的基数。由上述定义可以知道, 域 $D_1 \times D_2 \times \cdots D_n$ 上的关系 R , 就是由域 $D_1 \times D_2 \times \cdots D_n$ 确定的某些元组的集合。关系是笛卡儿的子积。

3.2.2 关系模式

由于关系实质上是一张二维表, 表的每一行称为一个元组, 每一列称为一个属性, 一个元组就是该关系所涉及的属性集的笛卡儿积的一个元素。关系是元组的集合, 因此关系模式 (Relation Schema) 要指出元组集合的结构。关系实际上就是关系模式在某一时刻的数据操作状态或内容。通常情况下把关系模式比作型, 而关系是它的值。因此现在把关系模式和关系统称为关系。一个关系模式应当是一个五元组, 关系模式可以形式化地表示为 $R(U, D, \text{DOM}, F)$, 其中:

- R 是关系名; U 是组成该关系的属性名集合; D 是属性组 U 中属性所来自的域; DOM 是属性值域的映像集合; F 是属性间的数据依赖关系集合。
- 例 3.2 将表 1.3.1 的学生关系通过 $R(U, D, \text{DOM}, F)$ 五元组的关系模式解释。
- R (关系名): 学生关系。
- U (属性集合): 学号、姓名、性别、年龄。
- D (域): 字符型 (学号、姓名、性别)、数值型 (年龄)。
- DOM (属性到域的映射): 学号 (字符型、宽度为 10); 姓名 (字符型、宽度为 8), 性别 (字符型、宽度为 2)、年龄 (整型)。
- F (属性间的数据依赖关系集合): (学号是唯一的, 能够分别决定姓名、性别、年龄)。

3.3 关系完整性

数据库的完整性指数据的正确性和相容性。与数据库的安全性不同, 数据库的完整性是为了防止数据库中存在不符合语义规定的数据、系统输入/输出无效信息, 同时还要使存储在不同副本中的同一个数据保持一致性而提出的。而安全性防范对象是非法用户和非法操作。维护数据库的完整性是数据库管理系统的基本要求。

为了维护数据库的完整性, 数据库管理系统 (DBMS) 必须提供一种机制来检查数据库

中的数据是否满足语义约束条件。这些加在数据库数据之上的语义约束条件称为数据库的完整性约束条件。DBMS 检查数据是否满足完整性约束条件的机制称为完整性检查。

3.3.1 完整性控制的含义

数据库的完整性包括数据的正确性、有效性和一致性。其中,正确性是指输入数据的合法性,例如,一个数值型数据只能有 0,1,⋯,9,不能含有字母和特殊字符,否则就不正确,失去了完整性。有效性是指所定义数据的有效范围,例如,人的性别不能有“男”“女”之外的值;人的一天最多工作时间不能超过 24 小时;工龄不能大于年龄等。一致性是指描述同一事实的两个数据应相同,例如,一个人不能有两个不同的性别、年龄等。

1. 数据库完整性控制的作用

数据库完整性控制对数据库系统是非常重要的,其作用主要体现在如下所述的 5 个方面。

(1) 数据库完整性约束能够防止合法用户使用数据库时向数据库中添加不符合语义规定的数据库。

(2) 利用基于 DBMS 的完整性控制机制来实现业务规划,易于定义,容易理解,而且可降低应用程序的复杂性,提高应用程序的运行效率。

(3) 基于 DBMS 的完整性控制机制是集中管理的,因此比应用程序更容易实现数据库的完整性。

(4) 合理的数据库完整性设计,能够同时兼顾数据库的完整性和系统的效能。

(5) 在应用软件的功能测试中,完善的数据库完整性有助于尽早发现应用软件的错误。

2. 对数据库完整性的破坏

通常情况下,对数据库的完整性破坏来自以下 6 个方面:

(1) 操作人员或终端用户的错误或疏忽。

(2) 应用程序(操作数据)错误。

(3) 数据库中并发操作控制不当。

(4) 由于数据冗余,引起某些数据在不同副本中不一致。

(5) DBMS 或者操作系统出错。

(6) 系统中任何硬件(如 CPU、磁盘、通道、I/O 设备等)出错。

数据库的数据完整性随时都有可能遭到破坏,应尽量减少被破坏的可能性,并且在数据遭到破坏后应能尽快地恢复到原样。因此,完整性控制是一种预防性的策略。完整性控制能够保证各个操作的结构得到正确的数据,即只要能确保输入数据的正确,就能够保证正确的操作产生正确的数据输出。

3.3.2 完整性约束条件

数据库的完整性是指数据的正确性和相容性。数据库是否具备完整性关系到数据库系统能否真实地反映现实世界,因此,维护数据库的完整性是非常重要的。

为维护数据库的完整性,DBMS 必须提供一种机制来检查数据库中的数据,看其是否满足语义规定的条件。这些加在数据库数据之上的语义约束条件称为数据库完整性约束条件,它们被作为模式的一部分存入数据库中。而 DBMS 中检查数据是否满足完整性条件的

机制称为完整性检查。

1. 完整性约束条件分类

完整性检查是围绕完整性约束条件进行的,因此完整性约束条件是完整性控制机制的核心。

完整性约束条件作用的对象可以是关系、元组、列三种。其中列约束主要是列的类型、取值范围、精度、排序等的约束条件。元组的约束是元组中各个字段间的联系的约束。关系的约束是若干元组间、关系集合以及关系之间的联系的约束。

完整性约束条件涉及的这三类对象,其状态可以是静态的,也可以是动态的。所谓静态约束是指数据库每一确定状态时的数据对象所应满足的约束条件,它是反映数据库状态合理性的约束,是最重要的一类完整性约束。动态约束是指数据库从一种状态转变为另一种状态时新、旧值之间所应满足的约束条件,它是反映数据库状态变迁的约束。

综上所述,我们可以将完整性约束条件分为 6 类。

(1) 静态列级约束。静态列级约束是对一个列的取值域的说明,这是最常用的,也是最容易实现的一类完整性约束,包括以下 3 个方面。

① 对数据类型的约束,包括数据的类型、长度、单位、精度等。例如,姓名类型为字符型,长度为 8;货物重量单位,使取值在正常范围内;性别的取值集合为(男,女),可以为性别字段创建 CHECK 约束。

② 对空值的约束。用 NOT NULL 来设定某列值不能为空。如果设定某列为 NOT NULL,则在添加记录时,此列必须插入数据。空值表示未定义或未知的值,与零值和空格不同。可以设置列不能为空值,例如,学生 ID 号不能为空值,而学生来自的省份可以为空值。

③ 对数据格式的约束,如规定日期的格式为 YYYY-MM-DD;对取值范围或取值集合的约束,如规定学生的成绩取值为 0~100。

(2) 静态元组约束。一个元组是由若干个列值组成的,静态元组约束就是规定元组的各个列之间的约束关系。

(3) 静态关系约束。在一个关系的各个元组之间或者若干关系之间常常存在各种联系或约束。常见的静态关系约束有:

① 实体完整性约束。

② 参照完整性约束。

实体完整性约束和参照完整性约束是关系模型的两个极其重要的约束,称为关系的两个不变性。

③ 函数依赖约束。大部分函数依赖约束都在关系模式中定义。

④ 统计约束。统计约束指字段值与关系中多个元组的统计值之间的约束关系。

(4) 动态列级约束。动态列级约束是修改列定义或列值时应满足的约束条件,包括以下两方面:

① 修改列定义时的约束。例如,将允许空值的列改为不允许空值时,如果该列目前已存在空值,则拒绝这种修改。

② 修改列值时的约束。修改列值有时需要参照其旧值,并且新旧值之间需要满足某种约束条件。例如,职工工资调整不得低于其原来工资,学生年龄只能增长等。

(5) 动态元组约束。动态元组约束是指修改元组的值时,元组中各个字段间需要满足某种约束条件。例如,职工工资调整时新工资不得低于原工资 $\times$ 工龄 $\times 1.5$,等等。

(6) 动态关系约束。动态关系约束是加在关系变化前后状态上的限制条件,例如事务一致性、原子性等约束条件。

2. 完整性控制

DBMS 的完整性控制机制应具有三个方面的功能。

(1) 定义功能:提供定义完整性约束条件的机制。

(2) 检查功能:检查用户发出的操作请求是否违背了完整性约束条件。

(3) 违约提示:如果发现用户的操作请求使数据违背了完整性约束条件,则采取一定的动作来保证数据的完整性。

3.3.3 完整性规则

为了实现对数据库完整性的控制,DBA 应向 DBMS 提出一组适当的完整性规则,这组规则规定用户在对数据库进行更新操作时,对数据检查什么,检查出错误后怎样处理等。

完整性规则规定了触发程序条件、完整性约束、违反规则的响应,即规则的触发条件,是指什么时候使用完整性规则进行检查;规则的约束条件,是指规定系统要检查什么样的错误;规则的违约响应,是指查出错误后应该怎样处理。

完整性规则是由 DBMS 提供,由系统加以编译并存放在系统数据字典中的,但在实际的系统中常常会省去某些部分。进入数据库系统后,就开始执行这些规则。这种方法的主要优点是违约响应所查出的错误由系统来处理,而不是让用户的应用程序来处理。另外,其规则集中存放在数据字典中,而不是散布在各个应用程序中,这样容易从整体上理解和修改。

3.3.4 实现参照完整性要考虑的问题

在关系系统中,最重要的完整性约束是实体完整性和参照完整性,其他完整性约束条件则可以归入用户定义的完整性中。目前 DBMS 中,提供了定义和检查实体完整性、参照完整性和用户定义完整性的功能。对于违反实体完整性和用户定义完整性的操作,一般拒绝执行,而对于违反参照完整性的操作,不是简单拒绝,而是根据语义执行一些附加操作,以保证数据库的正确性。下面详细讨论实现参照完整性要考虑的 4 个问题。

1. 外码能否接受空值问题

例如,学生表 Student、课程表 Course 和选课表 SC 中,Student 关系的主码为学号 sno, Course 关系的主码为 cno,SC 关系的主码为学号 sno 和 cno 组合码,外码为学号 sno 和 cno,称 SC 为参照关系,Student 和 Course 为被参照关系。

因此在实现参照完整性时,系统除了应该提供定义外码的机制外,还应提供定义外码列是否允许空值的机制。

2. 在被参照关系中删除元组的问题

一般地,当删除被参照关系的某个元组时,若参照关系存在若干元组,其外码值与被参照关系删除元组的主码值相同,这时可有三种不同的策略。

(1) 级联删除(Cascades)。将参照关系外码值与被参照关系中要删除元组主码值相同

的元组一起删除。

(2) 受限删除(Restricted)。仅当参照关系中没有任何元组的外码值与被参照关系中要删除元组的主码值相同时,系统才执行删除操作,否则拒绝此删除操作。

(3) 置空值删除(Nullifies)。删除被参照关系的元组,并将参照关系中相应元组的外码值置空值。

3. 在参照关系中插入元组时的问题

一般地,当参照关系插入某个元组,而被参照关系不存在相应的元组,其主码值与参照关系插入元组的外码值相同,这时可有以下策略。

(1) 受限插入。仅当被参照关系中存在相应的元组,其主码值与参照关系插入元组的外码值相同时,系统才允许插入,否则拒绝插入。

(2) 递归插入。首先向被参照关系中插入相应的元组,其主码值等于参照关系插入元组的外码值,然后向参照关系插入元组。

4. 修改关系中主码的问题

(1) 不允许修改主码。在有些 RDBMS 中,不允许修改关系主码。如上例中不能修改 Student 关系中的学号 sno。如果要修改,只能先删除,然后再增加。

(2) 允许修改主码。在有些 RDBMS 中,允许修改关系主码,但必须保证主码的唯一性和非空,否则拒绝修改。当修改的关系是被参照关系时,还必须检查参照关系。

从上面的讨论可以看到,DBMS 在实现参照完整性时,除了要提供定义主码、外码的机制外,还需要提供不同的策略供用户选择。选择哪种策略,都要根据应用环境的要求来确定。

3.3.5 完整性的定义

1. 实体完整性

实体完整性(Entity Integrity)约束:在关系数据库中一个关系对应现实世界的一个实体集,关系中的每一个元组对应一个实体。在关系中用主关键字来唯一标识一个实体,表明现实世界中的实体是可以相互区分、识别的,也即它们应具有某种唯一性来标识实体具有独立性,关系中的这种约束条件称为实体完整性。关系中的主键的特点是不能取“空”值,并且是唯一的,如表 1.3.1 中的“学号”就是主键。假如这个学号允许为空或是不唯一,则表中的记录将出现大量冗余或错误。例如,班级中有两个女生都叫“吴小晴”,年龄相同,如果学号允许为空,那么会出现两个一样的“吴小晴”,无法区别。

一个实体就是指表中的多条记录,而实体完整性是指在表中不能存在完全相同的两条或两条以上的记录,而且每条记录都要具有一个非空且不重复的主键值。

2. 域完整性

域完整性是指向表的某列添加数据时,添加的数据类型必须与该列字段数据类型、格式及有效的数据长度相匹配。通常情况下域完整性是通过 CHECK 约束、外键约束、默认约束、非空定义、规则以及在建表时设置的数据类型实现的。

3. 参照完整性

参照完整性(Referential Integrity)约束是定义建立关系之间联系的主关键字与外部关键字引用的约束条件。关系数据库中通常都包含多个存在相互联系的关系,关系与关系之

间的联系是通过公共属性来实现的。所谓公共属性 K ，理论上规定：若 K 是关系 S 中的一属性组，且 K 是另一关系 R 的主关键字，则称 K 为关系 S 对应关系 Z 的外关键字；若 K 是关系 S 的外关键字，则 S 中每一个元组在 K 上的值必须是空值或是对应关系 R 中某个元组的主关键字值。例如，有两个关系“系部”和“教师”，如表 1.3.7 表和表 1.3.8 所示。

表 1.3.7 系部表

系 号	系 名	联 系 电 话	院 长
01	机械系	666011	张明
02	管理系	666012	刘大海
03	计算机系	666013	王方

表 1.3.8 教师表

教 师 号	教 师 名	系 号
101	田一	01
102	赵海	02
103	李明	

参照完整性是指通过主键与外键建立两个或两个以上表的连接，建立连接的字段的数据类型和长度要保持一致。

4. 用户自定义完整性

实体完整性和参照完整性适用于任何关系型数据库系统，主要是针对关系的主关键字和外部关键字取值必须有效而设置的约束。用户自定义完整性(User-defined Integrity)约束则是根据应用环境的要求和实际需要，对某一具体应用所涉及的数据提出约束性条件。这一约束机制一般不应由应用程序提供，而应由关系模型提供定义并检验。用户定义完整性主要包括字段有效性和记录有效性。如表 1.3.7 所示，“系号”是字符型的，宽度为 2。还有表 1.3.1 的“性别”只能是“男”或“女”。

用户定义的完整性是根据具体的应用领域所要遵循的约束条件，由用户自己定义的特定的规则。

3.4 关 系 运 算

关系数据操作可以分为数据查询和数据更新两大类型，而关系运算是根据数据操作的需要提出来的。在关系操作中，以集合代数为基础运算的数据操作语言(DML)称为关系代数语言，关系和其上的关系代数运算组成一个代数，称为关系代数，关系代数是以为运算对象的一组高级运算的组合，是一种抽象的查询语言。

关系代数语言必须在查询表达式中标明操作的先后顺序，故表示同一结果的关系代数表达式可以有多种不同的形式。下面按照数据操作的两种类型分别介绍相应的关系代数运算。关系代数运算按运算符的不同可分为传统的集合运算和专门的关系运算两类。

3.4.1 传统的关系运算

传统的关系运算符包括并($\cup$)、差($-$)、交($\cap$)和笛卡儿积($\times$)4 种运算。设关系 R 和

关系 S 具有相同的 n 目属性,且相应的属性取自同一个域,则可以定义并、差、交运算。

1. 并

设关系 R 和关系 S 具有相同的目 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(Union)由属于 R 或属于 S 的元组组成。其结果关系仍为 n 目关系,如图 1.3.2 所示,记作

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

2. 差

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(Except)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,如图 1.3.3 所示,记作

$$Q = R - S = \{t \mid t \in R \text{ 但 } t \notin S\}$$

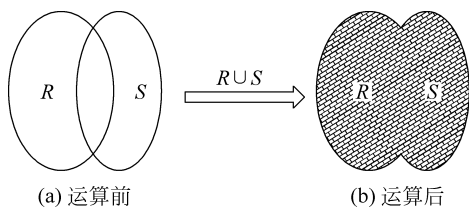


图 1.3.2 $R \cup S$ 的运算

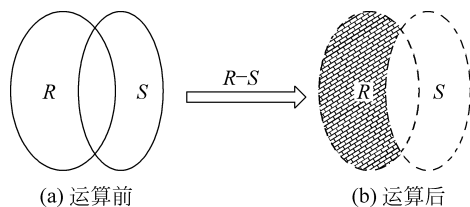


图 1.3.3 $R - S$ 的运算

3. 交

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(Intersection)由既属于 R 又属于 S 的元组组成。其结果关系仍为 n 目关系,如图 1.3.4 所示,记作

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

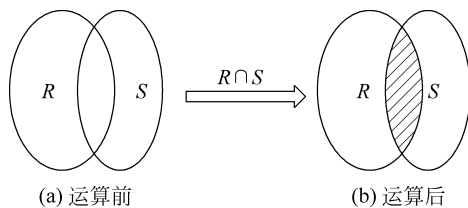


图 1.3.4 $R \cap S$ 的运算

4. 笛卡儿积

在这里的笛卡儿积是指广义的笛卡儿积(Extended Cartesian Product)。因为这里笛卡儿的元素是元组。广义笛卡儿积不要求参加运算的关系具有相同的目,广义笛卡儿积的运算是二元运算。

两个分别为 m 目和 n 目的关系 R 和关系 S 的广义笛卡儿积是一个有 $(m+n)$ 列的元组的集合。元组的前 m 个列是关系 R 的一个元组,后 n 个列是关系 S 的一个元组。若 R 有 K_1 个元组, S 有 K_2 个元组,则关系 R 和关系 S 的广义笛卡儿积有 $K_1 \times K_2$ 个元组,记作

$$R \times S = \{tq \mid t \in R \text{ and } q \in S\}$$

$R \times S$ 的结果是所有这样的元组对集合:一个元组来自 R ,另一个元组来自 S 。元组对

tq 表示将两个元组 t 和 q 连接起来得到的一个新元组。

图 1.3.5(f)所示为广义笛卡儿积运算的示意图。图 1.3.5(f) $R \times S$ 。 $R \times S$ 共有 $3+3$ 列,共有 4×4 个元组。

例 3.3 关系 R 和 S 分别具有三个属性,如图 1.3.5(a)、图 1.3.5(b)所示,关系 R 和 S 进行传统的集合运算的结果分别如图 1.3.5(c)~图 1.3.5(f)所示。

A	B	C
A ₁	B ₂	8
A ₂	B ₂	10
A ₃	B ₁	5
A ₂	B ₁	12

(a) 关系 R 的属性

A	B	C
A ₁	B ₂	8
A ₂	B ₁	12
A ₃	B ₃	6
A ₃	B ₁	5

(b) 关系 S 的属性

A	B	C
A ₁	B ₂	8
A ₂	B ₂	10
A ₃	B ₁	5
A ₂	B ₁	12
A ₃	B ₃	6

(c) $R \cup S$

A	B	C
A ₁	B ₂	8
A ₂	B ₁	12
A ₃	B ₁	5

(d) $R \cap S$

A	B	C
A ₂	B ₂	10

(e) $R - S$

A	B	C	A	B	C
A ₁	B ₂	8	A ₁	B ₂	8
A ₁	B ₂	8	A ₂	B ₁	12
A ₁	B ₂	8	A ₃	B ₃	6
A ₁	B ₂	8	A ₃	B ₁	5
A ₂	B ₁	10	A ₁	B ₂	8
A ₂	B ₁	10	A ₂	B ₁	12
A ₂	B ₁	10	A ₃	B ₃	6
A ₂	B ₁	10	A ₃	B ₁	5
A ₃	B ₃	5	A ₁	B ₂	8
A ₃	B ₃	5	A ₂	B ₁	12
A ₃	B ₃	5	A ₃	B ₃	6
A ₃	B ₃	5	A ₃	B ₁	5
A ₂	B ₁	12	A ₁	B ₂	8
A ₂	B ₁	12	A ₂	B ₁	12
A ₂	B ₁	12	A ₃	B ₃	6
A ₂	B ₁	12	A ₃	B ₁	5

(f) $R \times S$

图 1.3.5 传统的集合运算

3.4.2 专门的关系运算

专门的关系运算包括选择、投影、连接、除运算等。

1. 选择

选择(Selection)又称为限制(Restriction),它是在关系 R 中选择满足给定条件的元组,

组成一个新的关系,记作

$$\sigma_F(R) = \{t \mid t \in R \wedge F(t) = \text{'真'}\}$$

说明: 选择操作是根据某些条件对关系的水平切割,也就是从行的角度进行运算,选取符合条件的元组。选择运算如图 1.3.6 所示。

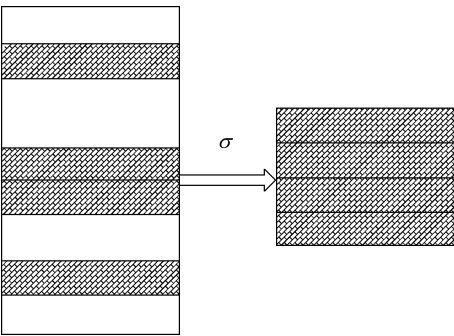


图 1.3.6 选择运算示意图

F 表示选择条件,是一个逻辑表达式,取值为逻辑“真”或“假”。逻辑表达式是由属性名(属性名也可以用它的列序号来代替)、常数(用引号括起来)、逻辑运算符($\neg$ 、 $\wedge$ 或 $\vee$,通常用 φ 表示逻辑运算符)、比较运算符($>$ 、 $\geq$ 、 $<$ 、 $\leq$ 、 $=$ 或 $<>$,通常用 θ 表示比较运算符)以及常用的函数(数学、字符、日期、转换等)组成。通常情况下,逻辑表达式是由逻辑运算符连接由比较运算符组成的比较关系式而成。

逻辑表达式 F 的基本形式为: $X_1\theta Y_1[\varphi X_2\theta Y_2]$, X_1 、 Y_1 等是属性名、常量或简单函数。

例 3.4 对关系 R (如图 1.3.7(a))进行相关的选择运算。求 $\sigma_{3>'6'}(R)$,从 R 中选择第 3 个分量值大于 6 的属性值,运算结果如图 1.3.7(b)所示。

A	B	C
A ₁	B ₂	8
A ₂	B ₁	6
A ₃	B ₂	5
A ₂	B ₁	10
A ₂	B ₁	7

(a) 关系 R

A	B	C
A ₁	B ₂	8
A ₂	B ₁	10
A ₂	B ₁	7

(b) $\sigma_{3>'6'}(R)$

图 1.3.7 例 3.4 的选择运算

2. 投影

投影(Projection)是指将对象转换为一种新形式的操作,该形式通常只包含那些随后将使用的属性列。通过投影,可在原来的关系上生成新的关系。也就是说投影运算是从给定关系的所有列中按某种顺序选取指定的列的集合,它是对数据库进行“纵向分割关系”的手

段,如图 1.3.8 所示,记为

$$\Pi_A(R) = \{t[A] \mid t \in R\}$$

其中: A 是属性名(即列名)表, R 是表名。

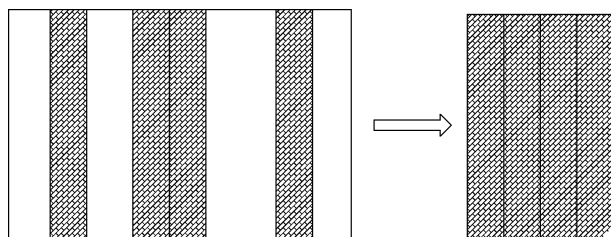


图 1.3.8 投影运算示意图

例 3.5 对例 3.4 的关系 R 进行相关的投影运算。(1)求 $\Pi_{A,B}(R)$, 运算结果如图 1.3.9(a)所示。(2)求 $\sigma_{3,1}(R)$, 即从 R 中选择第 3 个分量和第 1 个分量, 运算结果如图 1.3.9(b)所示。

A	B
A_1	B_2
A_2	B_1
A_3	B_2
A_2	B_1
A_2	B_1

(a) $\Pi_{A,B}(R)$

A	C
A_1	8
A_2	6
A_3	5
A_2	10
A_2	7

(b) $\sigma_{3,1}(R)$

图 1.3.9 例 3.5 的投影运算

3. 连接

虽然笛卡儿积可以实现两个或两个以上关系的乘积,但是新的关系数据冗余度大,系统费时太多。因此要能够得到简单而优化的新关系,对笛卡儿积进行限制,这就引入了连接(Join)运算。连接运算把投影运算和选择运算综合运用来解决复杂的数据库运算。

(1) θ 连接运算。 θ 连接运算是从两个已知关系 R 和 S 的笛卡儿积中,选取属性之间满足连接一定条件的元组组成新的关系,记作

$$R \bowtie_{A\theta B} S = \{t_r, t_s \mid t_r \in R \wedge t_s \in S \wedge t_r[A] \theta t_s[B]\}$$

其中 A 和 B 分别是关系 R 和 S 上度数相等且可比的属性组, θ 是比较运算符。连接运算从关系 R 和 S 的笛卡儿积 $R \times S$ 中选取 R 关系在 A 属性组上的值与 S 关系在 B 属性组上值满足比较关系 θ 的元组。

有两类常用的连接运算,一种是等值连接(Equijoin),另一种是自然连接(Natural Join)。

当 θ 为“=”时连接运算称为等值连接。它是从关系 R 和 S 的广义笛卡儿积中选取 A 、 B 属性值相等的元组。等值连接为

$$R \bowtie_{A=B} S = \{t_r, t_s \mid t_r \in R \wedge t_s \in S \wedge t_r[A] = t_s[B]\}$$

自然连接是特殊的等值连接,它要求两个关系中进行比较的分量必须是相同的属性,并且在结果中把重复的属性列去掉。即若 R 和 S 具有相同的属性组 B ,则自然连接可记作

$$R \bowtie S = \{ \widehat{t_r t_s} \mid t_r \in R \wedge t_s \in S \wedge t_r[B] = t_s[B] \}$$

一般连接运算是从行的角度进行运算的,但自然连接还需要去掉重复的列,所以同时从行和列的角度进行运算。

自然连接和等值连接的差别如下。

- 自然连接要求相等的分量必须有共同的属性名,等值连接则不要求。
- 自然连接要求把重复的属性名去掉,等值连接却不这样做。

例 3.6 关系 R 和 S 如图 1.3.10(a)、图 1.3.10(b) 所示,求 $R \bowtie_{C>D} S$ 。运算结果如图 1.3.10(c) 所示。

A	B	C
A ₁	B ₂	8
A ₂	B ₁	6
A ₃	B ₂	5
A ₂	B ₁	10
A ₂	B ₁	7

D	E
7	E ₁
6	E ₂
17	E ₃

A	B	C	D	E
A ₁	B ₂	8	7	E ₁
A ₂	B ₂	8	6	E ₂
A ₃	B ₁	10	7	E ₁
A ₂	B ₁	10	6	E ₂
A ₂	B ₁	7	6	E ₂

图 1.3.10 关系 R 、关系 S 及其连接运算 $R \bowtie_{C>D} S$

(2) F 连接运算。 F 连接运算是从关系 R 和 S 的笛卡儿积中选取属性值满足公式 F 的元组,设 F 为形如 $F_1 \wedge F_2 \wedge \cdots \wedge F_n$ 的公式,其中每个 $F_k (1 \leq k \leq n)$ 都是形如 $A \theta B$ 的算术比较式。这里 A 和 B 分别是关系 R 和关系 S 的第 A 个属性列名或序号、第 B 个属性列名或序号,记作

$$R \bowtie_F S = \sigma_F(R \times S)$$

例 3.7 关系 R 和 S 的 F 连接,如图 1.3.11 所示,这里 F 为 $R.B > S.D \wedge 3 > 1 (3 > 1$ 代表 R 关系中的第三列大于 S 中的第一列)。

在连接结果中会舍弃不满足连接条件的元组,这种形式的连接称为内连接。

如果希望不满足连接条件的元组也出现在连接结果中,可以通过外连接(Outer Join)操作实现。外连接是对自然连接运算的扩展。为什么要扩展自然连接?原因就是自然连接可能会造成信息丢失。所以为避免信息丢失,对失配元组,与一个空元组(所有属性值为 NULL)连接后,添加到结果关系中去,这就是外连接运算。

外连接有三种形式:左外连接(Left Outer Join)、右外连接(Right Outer Join)和全外连接(Full Outer Join)。

R			S	
A	B	C	D	E
A ₁	8	13	7	E ₁
A ₂	10	6	6	E ₂
A ₃	5	9	17	E ₃

A	B	C	D	E
A ₁	8	3	7	E ₁
A ₁	8	3	6	E ₂

图 1.3.11 F 连接结果 $R \bowtie S$ 关系
 $R.B > S.D \wedge 3 > 1$

$\bowtie$ 左外连接=自然连接+左侧表的失配元组(与空元组连接)

$\bowtie$ 右外连接=自然连接+右侧表的失配元组(与空元组连接)

$\bowtie$ 全外连接=自然连接+两侧表的失配元组(与空元组连接)

设有图 1.3.12(a)所示关系 R 和图 1.3.12(b)所示关系 S,则这两个关系的自然连接结果如图 1.3.12(c)所示,左外连接、右外连接、全外连接的结果如图 1.3.12(d)~(f)所示。

R			S		
A	B	C	B	C	D
a	1	a	2	c	2
b	3	c	1	a	1
c	5	e	5	e	4

R $\bowtie$ S			
A	B	C	D
a	1	a	1
c	5	e	4

R $\bowtie$ S			
A	B	C	D
a	1	a	1
c	5	e	4
b	3	c	NULL

R $\bowtie$ S			
A	B	C	D
a	1	a	1
c	5	e	4
c	5	e	4
NULL	2	c	2

R $\bowtie$ S			
A	B	C	D
a	1	a	1
c	5	e	4
b	3	c	NULL
NULL	2	c	2

图 1.3.12 连接运算举例

4. 除运算

除(Division)运算是指对于给定关系 $R(X,Y)$ 和 $S(Y,Z)$,其中 X,Y,Z 为单个属性或属性集, R 与 S 的除运算得到一个新的关系,它是 R 中满足下列条件的元组在 X 属性列上的投影:元组在 X 上分量值 x 的象集 Y_x 包含 S 在 Y 上投影的集合。记作

$$R \div S = \{t_r[X] \mid t_r \in R \wedge \Pi_Y(S) \subseteq Y_x\}$$

其中 Y_x 为 x 在 R 上的象集, $x = t_r[X]$ 。

关系 R 和关系 S 中的 Y 可以有不同的属性名,但是必须出自相同域集。

图 1.3.13 中, X_1 在 R 中的象集 $Z_{x_1} = \{Z_1, Z_2, Z_3\}$,

X_2 在 R 中的象集 $Z_{x_2} = \{Z_2, Z_3\}$,

X_3 在 R 中的象集 $Z_{x_3} = \{Z_1, Z_3\}$ 。

R	
X ₁	Z ₁
X ₁	Z ₂
X ₁	Z ₃
X ₂	Z ₂
X ₂	Z ₃
X ₃	Z ₁
X ₃	Z ₃

图 1.3.13 象集举例

例 3.8 设有关系 R 、 S 分别为图 1.3.14 中的(a)和(b), $R \div S$ 的结果为图 1.3.14(c)。
在关系 R 中, A 可以取 4 个值 $\{a_1, a_2, a_3, a_4\}$ 。其中:

a_1 的象集为 $\{(b_1, c_2), (b_2, c_3), (b_2, c_1)\}$

a_2 的象集为 $\{(b_3, c_7), (b_2, c_3)\}$

a_3 的象集为 $\{(b_4, c_6)\}$

a_4 的象集为 $\{(b_6, c_6)\}$

S 在 (B, C) 上的投影为 $\{(b_1, c_2), (b_2, c_3), (b_2, c_1)\}$

显然只有 a_1 的象集 $(B, C)_{a_1}$ 包含了 S 在 (B, C) 属性组上的投影, 所以

$$R \div S = \{a_1\}$$

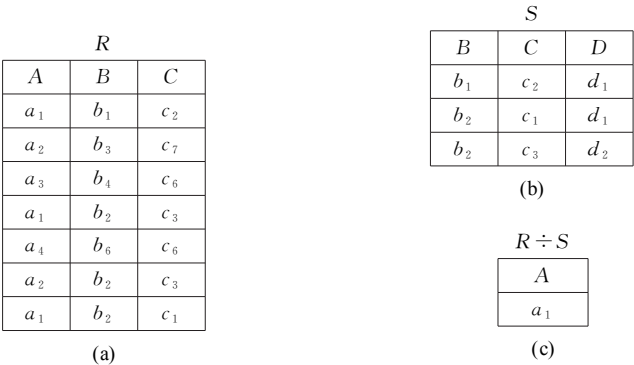


图 1.3.14 除运算举例

5. 关系运算综合实例

例 3.9 设有一个学生缴费管理库, 包括学生信息关系(Studenta)、缴费关系(JF)、部门信息关系(Group), 如图 1.3.15(a)、图 1.3.15(b)、图 1.3.15(c)所示。

(1) 查询一下山东省有哪些学生?

$$\sigma_{\text{Stusheng} = \text{'山东省'}}(\text{Studenta})$$

结果为:

Stuaid	Stuname	Stusex	Stuage	Stumz	Stusheng	Zgid
10001001	刘晓那	女	20	汉族	山东省	信息
10001004	赵海威	男	20	汉族	山东省	工商

(2) 查询学生的学号和姓名。

$$\Pi_{\text{Stuaid}, \text{Stuname}}(\text{Studenta})$$

结果为:

Stuaid	Stuname
10001001	刘晓那
10001002	冯刚
10001003	李佳
10001004	赵海威
10001005	王梅

Studenta						
学号 Stuaid	姓名 Stuname	性别 Stusex	年龄 Stuage	民族 Stumz	省份 Stusheng	专业 Zgid
10001001	刘晓那	女	20	汉族	山东省	信息
10001002	冯刚	男	21	汉族	吉林省	信息
10001003	李佳	女	22	满族	辽宁省	机械
10001004	赵海威	男	20	汉族	山东省	工商
10001005	王梅	女	22	满族	吉林省	机械

(a) 学生信息关系

JF		
部门(id)Gid	部门名称 Gname	部门类型 Gclass
0001	财务处	教辅
0002	教务处	教学
0003	后 勤	教辅

(b) 缴费关系

Group		
新生(id)Stuaid	部门 id Gid	缴费金额 JF
10001001	0001	3456
10001001	0002	500
10001001	0003	600
10001002	0001	3456
10001002	0002	500
10001002	0003	600
10001003	0003	600
10001005	0001	3456

(c) 部门信息关系

图 1.3.15 学生缴费管理库

(3) 查询没有缴费的学生姓名。

$$\Pi_{\text{Stuname}}(\sigma_{\text{Gname}='后勤'}(\text{JF}) \bowtie \Pi_{\text{Stuaid}, \text{Stuname}}(\text{Studenta}))$$

结果为:

Stuname

 刘晓那
 冯刚
 李佳

例 3.10 设有关系 Group 为 R(如图 1.3.15(c)所示),和关系 S₁(如图 1.3.16(a)所示)、关系 S₂(如图 1.3.17(a)所示)和关系 S₃(如图 1.3.18(a)所示)相除。

(1) 查询 0001 部门缴费的学生名单,关系 S₁ 如图 1.3.16(a)所示,R ÷ S₁ 结果如图 1.3.16(b)所示。

(2) 查询 Stuaid(新生 id)为 10001 和 Gid(部门 id)为 0002 的已缴费的学生的名单,S₂ 关系如图 1.3.17(a)所示,R ÷ S₂ 结果如图 1.3.17(b)所示。

Gid(部门 id)
0001

(a) 关系 S_1

Stuaid(新生 id)	JF 缴费金额
10001	3456
10002	3456
10005	3456

(b) $R \div S_1$ 结果

图 1.3.16 关系操作

Stuaid(新生 id)	Gid(部门 id)
10001	0002

(a) 关系 S_2

Stuaid(新生 id)	JF 缴费金额
10001	500

(b) $R \div S_2$ 结果

图 1.3.17 关系操作

(3) 查询 0001、0002、0004 部门已缴费的学生的名单,关系 S_3 如图 1.3.18(a)所示, $R \div S_3$ 结果如图 1.3.18(b)所示。

Gid(部门 id)
0001
0002
0004

(a) 关系 S_3

Stuaid(新生 id)
10001

(b) $R \div S_3$ 结果

图 1.3.18 关系操作

3.5 复习思考

3.5.1 小结

本章主要介绍了关系数据模型是以集合论中的关系概念为基础发展起来的。关系模型中无论是实体还是实体间的联系均由单一的结构类型——关系来表示。在实际的关系数据库中的关系也称表。

以关系模型为基础,介绍了关系数据库理论以及关系模型数据结构: 关系、元组、属性、主码、域、分量等; 关系完整性: 实体完整性约束(实体完整性、参照完整性和用户定义的完整性)。

关系数据库中传统的关系运算,并、交、差和笛卡儿积运算以及专门的关系运算包括选择、投影、连接和除运算。

3.5.2 习题

一、选择题

1. 概念模型是现实世界的第一层抽象,这一类模型中常用的模型是()。
A. 层次模型
B. 关系模型
C. 网状模型
D. 实体-联系模型
2. 区分不同实体的依据是()。
A. 名称
B. 属性
C. 对象
D. 概念

3. 下面的选项不是关系数据库基本特征的是()。
 - A. 不同的列应有不同的数据类型
 - B. 不同的列应有不同的列名
 - C. 与行的次序无关
 - D. 与列的次序无关
4. 一个关系只有一个()。
 - A. 候选码
 - B. 外码
 - C. 超码
 - D. 主码
5. 关系模型中,一个码是()。
 - A. 可以由多个任意属性组成
 - B. 至多由一个属性组成
 - C. 由一个或多个属性组成,其值能够唯一标识关系中一个元组
 - D. 以上都不是
6. 关系数据库管理系统应能实现的专门关系运算包括()。
 - A. 排序、索引、统计
 - B. 选择、投影、连接
 - C. 关联、更新、排序
 - D. 显示、打印、制表
7. 4 种基本关系代数运算是()。
 - A. 并($\cup$)、差($-$)、选择(σ)和笛卡儿积($\times$)
 - B. 投影(Π)、差($-$)、交($\cap$)和笛卡儿积($\times$)
 - C. 投影(Π)、选择(σ)、除($\div$)、差($-$)
 - D. 并($\cup$)、差($-$)、交($\cap$)和笛卡儿积($\times$)
8. 从一个数据库文件中取出满足某个条件的所有记录形成一个新的数据库文件的操作是()操作。
 - A. 投影
 - B. 连接
 - C. 选择
 - D. 复制
9. 关系代数中的连接操作是由()操作组合而成。
 - A. 选择和投影
 - B. 选择和笛卡儿积
 - C. 投影、选择、笛卡儿积
 - D. 投影和笛卡儿积
10. 一般情况下,当对关系 R 和 S 进行自然连接时,要求 R 和 S 含有一个或者多个共有的()。
 - A. 记录
 - B. 行
 - C. 属性
 - D. 元组
11. 假设有关系 R 和 S ,关系代数表达式 $R - (R - S)$ 表示的是()。
 - A. $R \cap S$
 - B. $R \cup S$
 - C. $R - S$
 - D. $R \times S$

二、填空题

1. 1970 年,IBM 公司的研究员_____博士发表的《大型共享数据库的关系模型》一文提出了关系模型的概念。
2. 在关系模型中,基本包括_____,_____,_____,_____,_____分量等。
3. 关系代数运算中,基本的运算是_____,_____,_____,_____。
4. 关系代数运算中,专门的关系运算有_____,_____,_____。
5. 关系代数中,从两个关系中找出相同元组的运算称为_____运算。
6. 设有学生关系: $S(Sno, Sname, Ssex, Sage, Sl)$ 。在这个关系中, Sno 表示学号, $Sname$ 表示姓名, $Ssex$ 表示性别, $Sage$ 表示年龄, Sl 表示寝室。则查询学生姓名和所在寝

室的投影操作的关系运算式是_____。

三、简答题

- 1. 解释下述名词概念：关系模型、关系、属性、域、元组、候选码、码、实体完整性规则、参照完整性规则。
- 2. 为什么关系中的元组没有先后顺序？
- 3. 为什么关系中不允许有重复的元组？
- 4. 等值连接、自然连接二者之间有什么区别？
- 5. 已知关系 R 和 S ，分别计算① $R \cup S$ ，② $R \cap S$ ，③ $R - S$ ，④ $R \times S$ 。

R		
A	B	C
a_1	b_1	6
a_1	b_2	8
a_2	b_3	9
a_3	b_3	6

S		
A	B	C
a_1	1	7
a_2	b_2	8
a_3	b_4	6