

数据库系统原理及应用 (第 4 版)

关系模型与关系代数

Gavin-Yi LIU

<https://github.com/songzitea>

<https://gitee.com/songzitea>

Knowledge Artificial Intelligence(KAI)



1. 回顾

2. 关系模型

- 2.1. 关系数据结构
- 2.3. 关系完整性约束
- 2.4. 关系操作

3. 关系运算

- 3.1. 关系代数
 - 传统的集合运算
 - 专门的关系运算
 - 关系代数查询综合
- 3.2. 关系演算
 - 元组关系演算
 - 域关系演算

4. 参考文献

1. 回顾

2. 关系模型

- 2.1. 关系数据结构
- 2.3. 关系完整性约束
- 2.4. 关系操作

3. 关系运算

- 3.1. 关系代数
 - 传统的集合运算
 - 专门的关系运算
 - 关系代数查询综合
- 3.2. 关系演算
 - 元组关系演算
 - 元组关系演算

4. 参考文献

数据模型（回顾）

- 数据模型是一个描述**数据结构**、**数据操作**以及**数据约束**的数学形式体系（即概念及其符号表示系统）
- 根据**数据抽象**的不同级别，将数据模型划分为 3 类：
 - 概念模型：概念层次的数据模型，也称为信息模型
 - 逻辑模型：用于描述数据库数据的整体逻辑结构
 - 物理模型：用来描述数据的物理存储结构和存取方法
- 不同的 DBMS 提供不同的逻辑数据模型：
 - 层次模型 (hierarchical model)
 - 网状模型 (network model)
 - 关系模型 (relational model)——本章的内容

Outline

1. 回顾

2. 关系模型

- 2.1. 关系数据结构
- 2.3. 关系完整性约束
- 2.4. 关系操作

3. 关系运算

- 3.1. 关系代数
 - 传统的集合运算
 - 专门的关系运算
 - 关系代数查询综合
- 3.2. 关系演算
 - 元组关系演算
 - 域关系演算

4. 参考文献

- 系统而严格地提出关系模型的是美国 IBM 公司的 E.F.Codd 1970 年提出关系数据模型 [1]。
- 关系数据库系统是支持关系数据模型的数据库系统。关系数据库管理系统是当今的主流数据库管理系统。
- 关系模型由关系数据结构、关系操作集合和关系完整性约束三部分组成。

关系数据模型

- 关系数据结构
 - 如何描述真实世界的实体
 - 如何描述实体之间的关系
- 完整性约束规则
 - 数据需遵循的规则
 - 联系需遵循的规则
- 运算符集合
 - 操纵数据
 - 支持查询

Outline

1. 回顾

2. 关系模型

2.1. 关系数据结构

2.3. 关系完整性约束

2.4. 关系操作

3. 关系运算

3.1. 关系代数

传统的集合运算

专门的关系运算

关系代数查询综合

3.2. 关系演算

元组关系演算

元组关系演算

4. 参考文献

关系模型的基本概念---关系

- 关系模型的数据结构非常简单，它就是二维表，亦称为关系（Relation）。通俗地讲，即是由行和列组成的二维表，二维表的名字就是关系的名字。
- 关系数据库是表的集合，即关系的集合。表是一个实体集，一行就是一个实体，它由共同表示一个实体的有关联的若干属性的值所构成。
- 由于一个表是这种有关联的值的集合（即行的集合），而表这个概念和数学上的关系概念密切相关，因此称为关系模型。
- 关系模型中，现实世界的实体以及实体间的各种联系都是用关系来表示，如表1中关系的名字就是教师实体集。

Table 1: 教师实体集

TeacherNo	TeacherName	Sex	Prof.	DeptName	Scale
22001	赵乾	女	讲师	计算机	6000
22002	孙震	女	讲师	自动化	6000
22003	李勇	男	教授	计算机	8000
22004	李离	女	教授	通信	8000
22005	钱坤	男	副教授	自动化	7000

关系模型的基本概念---关系的性质

关系可以看成是二维表，但并不是所有的二维表都是关系。在关系模型中，对关系做了一定的限制。总的来说，关系具有如下性质。

- 列是同质的 (Homogeneous)，即每一列中的分量必须来自同一个域且必须是同一类型的数据。
- 不同的属性可来自同一个域，但不同的属性有不同的名字。其中的每一列称为一个属性，不同的属性要给予不同的属性名。
- 列的顺序可以任意交换，但交换时应连同属性名一起交换，否则将得到不同的关系。
- 元组 (行) 的顺序可任意交换。在关系数据库中，可以按照各种排序要求对元组的次序重新排列。
- 关系中不允许出现相同的元组。关系中的一个元组表示现实世界中的一个实体或一个实体间的联系，如果元组重复则表示实体或实体间的联系重复，这样不仅会造成数据库中数据的冗余，也可能造成数据查询与统计的结果出现错误。
- 关系中的每一个分量必须是不可再分的数据项，即所有属性值都是一个单独的值，而不是值的集合。

关系模型的基本概念---域与笛卡尔积

- 域 (Domain)是一组具有相同数据类型的值的集合。
- 笛卡尔积 (Cartesian Product)给定一组域 D_1, D_2, \dots, D_n , 它们之中可以有相同的域。 D_1, D_2, \dots, D_n 的笛卡尔积为:

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

其中:

- 集合中的每一个元素 (d_1, d_2, \dots, d_n) 称为一个**n 元组**, 简称为**元组**(Tuple); 元素中的每一个值 d_i 称为一个**分量**(Component)。
- 一个域允许的不同取值的个数称为这个域的**基数**
- 若 $D_i(i = 1, 2, \dots, n)$ 为有限集, 假设其**基数**为 $m_i(i = 1, 2, \dots, n)$, 则 D_1, D_2, \dots, D_n 的**基数 M** 为:

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

关系模型的基本概念---域与笛卡尔积

Example

给定两个域：

- 学生的姓名集合： $D_1 = \{ \text{'李小勇'}, \text{'刘方晨'}, \text{'王红敏'} \}$
- 课程的名称集合： $D_2 = \{ \text{'数据库系统概论'}, \text{'操作系统'} \}$

则 D_1, D_2 的笛卡尔积

$$D_1 \times D_2 = \{ (\text{'李小勇'}, \text{'数据库系统概论'}), \\ (\text{'李小勇'}, \text{'操作系统'}), \\ (\text{'刘方晨'}, \text{'数据库系统概论'}), \\ (\text{'刘方晨'}, \text{'操作系统'}), \\ (\text{'王红敏'}, \text{'数据库系统概论'}), \\ (\text{'王红敏'}, \text{'操作系统'}) \}$$

Table 2: D_1 与 D_2 的笛卡尔积

姓名	课程名称
李小勇	操作系统
刘方晨	数据库系统概论
刘方晨	操作系统
王红敏	数据库系统概论
王红敏	操作系统

关系模型的基本概念---属性

- D_1, D_2, \dots, D_n 的子集称为在域 D_1, D_2, \dots, D_n 上的关系，表示为：

$$r(D_1, D_2, \dots, D_n)$$

其中， r 表示关系的名字， n 是关系的**目或度**(degree)。

- 当 $n=1$ 时，该关系称为**单元关系**；
- 当 $n=2$ 时，称为**二元关系**。

- **关系**是笛卡尔积的**有限子集**，所以关系也是一个二维表，表的每行对应于关系的一个元组，表的每列对应于关系的一个域。
- 由于域可以相同，为了区别就必须给每列起一个名字，称为**属性** (attribute)。 n 目关系共有 n 个属性。

关系模型的基本概念-空值

- 空值 (null) 是所有可能的域的一个取值，表明值未知或值不存在。
 - 对于学位的取值域，某员工的学位为空值 null，表示不知道该员工所获得的学位，或该员工没有获得学位；
 - 对于成绩的取值域，某学生的成绩为空值 null，表示不知道该学生的成绩，或该学生没有成绩（如没有参加考试就没有获得成绩）。
 - 在进行关系操作时，有时关系中的某属性值在当前是填不上的，比如档案中有“生日不详”、“下落不明”、“日程尚待公布”等，这时就需要空值来代表这种情况。关系模型中用 ‘?’ 表征

空值的含义

- 数据库中有了空值，会影响许多方面，如影响聚集函数运算的正确性，不能参与算术、比较或逻辑运算等
- 例如：“ $3 + ?$ ” 结果是多少呢？ “ $3 * ?$ ” 结果是多少呢？ “ $? \text{ and } (A=A)$ ” 结果又是多少呢？
- 再例如，一个班有 30 名同学，如所有同学都有成绩，则可求出平均成绩；如果有一个同学没有成绩，怎样参与平均成绩的计算呢，是当作 0，还是当作 100 呢？还是不考虑他呢？
- 有空值的时候是需要特殊处理的，要特别注意。

关系模型的基本概念-空值

Table 3: 学生关系 Student

studentNo	studentNo	sex	birthday	speciality
2101001	李小勇	男	2003-12-21	计算机
2101008	王红	男	2005-04-26	计算机
2102002	刘方晨	女	2003-11-11	NULL
2102005	王红敏	女	2003-10-01	信息系统
2203045	王红	男	2005-04-26	会计学
2203010	李宏冰	女	2005-03-09	NULL

关系的最基本要求：

- 关系中的每个属性的域必须是原子的，即域中的每个值都是不可再分的一个完整单元。
- 关系中的每个元组都是可区分的，即存在唯一标识不同元组的属性(集)——码。

关系模型的基本概念---关系模式

- 对于一个二维表，有表头部分和表体部分：
 - 表头部分定义了该表的结构，即定义了该表由哪些列构成（假设由 n 列构成），每个列的名字和取值范围等；
 - 表体部分就是所有数据行（元组）的集合，每一个数据行都是由表头部分规定的 n 列有关联的取值的集合构成。
- 表体部分对应于关系，
 - 每一个数据行对应于关系的一个元组，即关系是元组的集合。
 - 关系是值的概念。
- 表头部分对应于关系模式，它定义了元组集合的结构，
 - 即定义了一个元组由哪些属性构成（假设由 n 个属性构成），每个属性的名字和来自的域等。
 - 关系模式是型的概念。

关系模型的基本概念---关系模式

- 关系的描述称为关系模式 (relation schema), 形式化地表示为:

$r(U, D, DOM, I, F)$

- r 为关系名,
- U 为组成该关系的属性名的集合,
- D 为属性集 U 中所有属性所来自的域的集合,
- DOM 为属性向域的映像集合,
- I 是完整性约束集合, 用于确保数据的正确性和一致性。
- F 为属性间数据的依赖关系集合 (即体现一个元组的各属性取值之间的“关联”性)。

- 关系模式通常被简记为: $r(U)$ 或 $r(A_1, A_2, \dots, A_n)$ r 为关系名, U 为属性名的集合 $\{A_1, A_2, \dots, A_n\}$

关系模式表示例子

假设我们有一个学生信息表，用于存储学生的姓名、年龄和成绩。我们将通过关系模式的形式化表示来描述这个表。

- 关系名 (r)，表示这个关系的名称。例如，我们把这个表命名为 Student。
- 属性名集合 (U)：表示组成该关系的属性名的集合。在这个例子中，
 $U = \{\text{学号, 姓名, 年龄, 成绩}\}$
- 域集合 (D)：表示属性集 U 中所有属性所来自的域的集合。在这个例子中，
 - 学号：整数类型，范围是 [100000, 999999]。
 - 姓名：字符串类型，长度不超过 50 个字符。
 - 年龄：整数类型，范围是 [0, 100]。
 - 成绩：浮点数类型，范围是 [0.0, 100.0]。

因此，域集合 D 可以表示为： $D = \{\text{整数, 字符串, 整数, 浮点数}\}$

关系模式表示例子 (CONT.)

- 属性向域的映射集合 (DOM): 表示属性向域的映射集合, 即每个属性对应的取值范围。在这个例子中,

$$DOM = \left\{ \begin{array}{l} \text{学号} \rightarrow \text{整数范围}[100000, 999999], \\ \text{姓名} \rightarrow \text{字符串长度} \leq 50, \\ \text{年龄} \rightarrow \text{整数范围}[0, 100], \\ \text{成绩} \rightarrow \text{浮点数范围}[0.0, 100.0] \end{array} \right\}$$

- 完整性约束集合 (I): 表示完整性约束集合, 用于确保数据的正确性和一致性。在这个例子中,

$$I = \left\{ \begin{array}{l} \text{学号是主键}, \\ \text{年龄} \in [0, 100], \\ \text{成绩} \in [0.0, 100.0] \end{array} \right\}$$

- 数据依赖关系集合 (F): 表示属性间数据的依赖关系集合, 即体现一个元组的各属性取值之间的“关联”性。在这个例子中,

$$F = \left\{ \begin{array}{l} \text{学号} \rightarrow \text{姓名}, \\ \text{学号} \rightarrow \text{年龄}, \\ \text{学号} \rightarrow \text{成绩} \end{array} \right\}$$

关系模式表示例子 (CONT.)

将上述所有部分组合起来，学生信息表的关系模式可以表示为： $\text{Student}(U, D, \text{DOM}, I, F)$ 具体展开为：

$$\text{Student} \left\{ \begin{array}{l} \{\text{学号, 姓名, 年龄, 成绩}\}, \{\text{整数, 字符串, 整数, 浮点数}\}, \\ \left\{ \begin{array}{l} \text{学号} \rightarrow \text{整数范围}[100000, 999999], \\ \text{姓名} \rightarrow \text{字符串长度} \leq 50, \\ \text{年龄} \rightarrow \text{整数范围}[0, 100], \\ \text{成绩} \rightarrow \text{浮点数范围}[0.0, 100.0] \end{array} \right\}, \\ \left\{ \begin{array}{l} \text{学号是主键,} \\ \text{年龄} \in [0, 100], \\ \text{成绩} \in [0.0, 100.0] \end{array} \right\}, \left\{ \begin{array}{l} \text{学号} \rightarrow \text{姓名}, \\ \text{学号} \rightarrow \text{年龄}, \\ \text{学号} \rightarrow \text{成绩} \end{array} \right\} \end{array} \right\}$$

关系的码

- **超码**: 对于关系 r 的一个或多个属性的集合 A , 如果属性集 A 可以唯一地标识关系 r 中的一个元组, 则称属性集 A 为关系 r 的一个**超码 (superkey)**。
- **候选码**: 对于关系 r 的一个或多个属性的集合 A , 如果属性集 A 是关系 r 的超码, 且属性集 A 的**任意真子集都不能成为关系 r 的超码**, 则称属性集 A 为**候选码 (candidate key)**。候选码具有唯一性、最小性的特点。
- **主码**: 若一个关系有多个候选码, 则可以选定其中的一个候选码作为该关系的**主码**。
- **外码**: 设 F 是关系 r 的一个属性 (或属性集), K_s 是关系 s 的**主码**。如果 F 与 K_s 相对应 (**即关系 r 中属性 F 的取值范围对应于关系 s 中主码 K_s 的取值范围的子集**), 则称 F 是关系 r 参照关系 s 的**外码 (foreign key)**, 简称 F 是关系 r 的外码。称关系 r 为参照关系, 关系 s 为被参照关系或目标关系。



Figure 1: 外码参照图

主属性与非主属性

- 包含在任一候选码中的属性称为主属性，不包含在任一候选码中的属性称为非主属性。
 - 例如，在没有重名学生的情况下，
■ 学生关系的属性“学号”与“姓名”都是学生关系的候选码，则它们都是学生关系的主属性。
■ 而属性“性别”与“系别”不包含在任一候选码中，则它们都是学生关系的非主属性。
- 在最简单的情况下，关系的候选码只包含一个属性；在最极端的情况下，关系的候选码是所有属性的组合，这时称为全码。
 - 例如，设有关系演出（演奏者编号，乐器编号，演播室编号），其中的3个属性分别为演奏者关系、乐器关系及演播室关系的主码，它们共同唯一标识了一个演出，则演出关系的主码为它们的组合，即为全码。

关系数据库模式

- 关系数据库也有型和值之分：
 - 型就是关系数据库模式，即它所包含的所有关系模式的集合；
 - 值就是这些关系模式在某一时刻所对应的关系的集合，通常就称为关系数据库实例。
- 在实际应用中，人们经常把关系数据库模式和关系数据库实例都笼统地称为关系数据库。

关系、关系模式与关系实例

- 关系是一个数据集合
- 关系模式描述关系的数据结构和语义约束的集合
- 关系模式是相对稳定的
- 关系实例是随时间而变化的
- 关系实例是某一时刻现实世界状态的真实反映

关系、关系模式与关系实例

- 关系数据库模式: 一组关系模式的集合 $r = \{A_1, A_2, \dots, A_n\}$, 其中 A_i 是第 i 个关系模式。
- 关系数据库实例: 关系数据库模式 $r = \{A_1, A_2, \dots, A_n\}$ 对应的关系数据库实例是一组关系实例的集合 $r = \{I_1, I_2, \dots, I_n\}$ 其中 I_i 是 A_i 的关系实例

Example (关系数据库模式)

```
r={Student(姓名, 学生编号, 年级, 专业, 系),  
    Course(课程名, 课程编号, 学分, 系),  
    Courseteaching(系, 专业, 课程编号, 学期, 年, 教师),  
    Prerequest(课程编号, 前序课程编号),  
    Grade(学生编号, 课程编号, 成绩)}
```

1. 回顾

2. 关系模型

2.1. 关系数据结构

2.3. 关系完整性约束

2.4. 关系操作

3. 关系运算

3.1. 关系代数

传统的集合运算

专门的关系运算

关系代数查询综合

3.2. 关系演算

元组关系演算

元组关系演算

4. 参考文献

完整性约束

- 为了维护关系数据库中数据与现实世界的一致性，关系数据库中数据的插入、删除和修改操作必须遵守一定的约束条件，即**关系模型的完整性约束**。
- 关系模型中有 3 类完整性约束，即**实体完整性**、**参照完整性**与**用户定义完整性**。
- 实体完整性和参照完整性**是关系模型必须满足的**完整性约束条件**，称为关系的**两个不变性**，任何关系数据库系统都支持这两类完整性约束。

Definition (实体完整性)

若属性集 A 是关系 r 的主码，则 A 不能取空值 $null$ 。

- 关系 Student，由于 studentNo 是关系 Student 的主码，因此它在任何时候的取值都不能为空值 null。如果主码是由若干个属性的集合构成，则要求构成主码的每一个属性的值都不能取空值。
- 关系 Score，它的主码是 studentNo, courseNo, term，因此这 3 个属性都不能取空值 null。
- 关系 Course 选课 (StudentNo,CourseNo,Score) 中，主码为 (StudentNo,CourseNo)，则“StudentNo”与“CourseNo”两个属性值都不能为空。

参照完整性

Definition (参照完整性)

若属性 (或属性集) F 是关系 r 的外码, 它与关系 s 的主码 K_s 相对应, 则对于关系 r 中的每一个元组在属性 F 上的取值要么为空值 $null$, 要么等于关系 s 中某个元组的主码值。

已知学生、课程与选课的关系如下:

学生 (学号, 姓名, 性别, 系别, 年龄)

课程 (课程编号, 课程名称, 学分)

选课 (学号, 课程编号, 成绩)

通过分析可知,

- 学生关系的主码为“学号”, 课程关系的主码为“课程编号”, 选课关系的主码为(学号, 课程编号), 而属性“学号”与“课程编号”又是选课关系的外码,
- 所以, 选课关系中的属性“学号”与“课程编号”只能取学生关系中属性“学号”与课程关系中属性“课程编号”中已经存在的值 (根据实体完整性规则主码不可为空)。

参照完整性: 实现一(多)对一(多)联系的外码

学生成绩管理数据库 ScoreDB 中, 学生关系 Student 与班级关系 Class 之间存在多对一的“归属”联系, 如图2所示。假设每一个学生一个学期可以选修若干门课程, 每一门课程同时有若干个学生选修, 那么学生关系 Student 与课程关系 Course 之间存在多对多的“选修”联系, 且课程的开课学期 term 和成绩 score 为联系属性, 如图3所示。

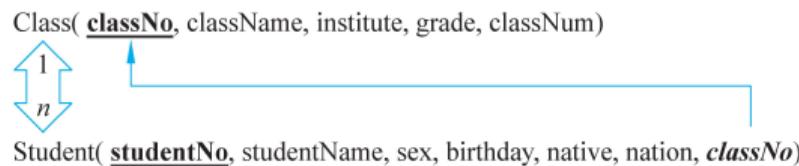


Figure 2: 实现“多对一”联系的外码

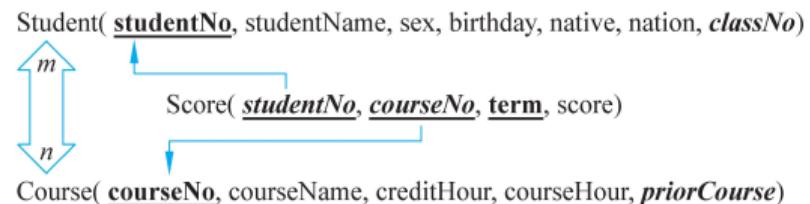


Figure 3: 实现“多对多”联系关系及外码

假设一门课程可能存在先修课程, 且关系 Course 中的 priorCourse 属性用来存放先修课程的课程编号。属性 priorCourse 是课程关系 Course 参照课程关系 Course 的外码, 如图4所示。

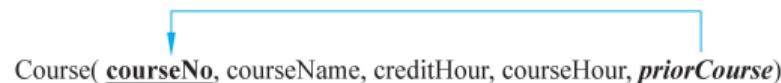


Figure 4: 实现关系内部多对一联系的外码

数据库模式导航图

数据库模式导航图：一个含有主码和外码依赖的数据库模式可以通过模式导航图来表示，如图5所示。

- 关系 Student 与 Class 之间存在多对一的“归属”联系（一个班由多个学生组成，一个学生只能归属与某个班），通过外码 classNo 实现该联系。
- 关系 Student 与 Course 之间存在多对多的“选修”联系。
- 关系 Score 的主码是 studentNo, courseNo, term，显然同一个学生在同一个学期不允许修读同一门课程多次。
- studentNo、courseNo 都是关系 Score 的外码，分别实现与关系 Student、Course 之间的多对一联系（间接地实现了关系 Student 与 Course 之间的多对多“选修”联系）。
- 关系 Course 的外码 priorCourse 参照本关系的主码 courseNo。

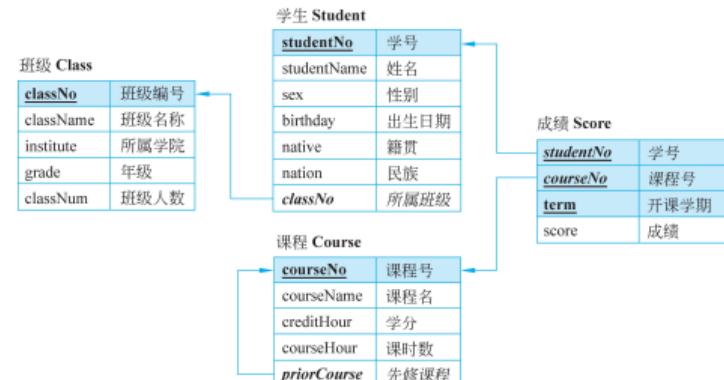


Figure 5: ScoreDB 的模式导航图

用户自定义完整性

- 任何关系数据库管理系统都应该支持**实体完整性和参照完整性**。
- **用户定义完整性**是针对某一具体应用要求来定义的约束条件，它反映某一具体应用所涉及的数据必须满足的语义要求。
- 关系模型应提供定义和检验这类完整性的机制，以便用统一的、系统的方式处理它们，而不须由应用程序承担这一功能。例如：
 - 限制关系中某些属性的取值要符合业务语义要求。规定选课关系中成绩属性的取值范围为0-100之间的整数，或限制某些数据的输入格式等。
 - 限制关系中某些属性的取值之间需要满足一定的逻辑关系。
 - 限制关系中某属性集上的取值必须唯一。

Outline

1. 回顾

2. 关系模型

- 2.1. 关系数据结构
- 2.3. 关系完整性约束
- 2.4. 关系操作

3. 关系运算

- 3.1. 关系代数
 - 传统的集合运算
 - 专门的关系运算
 - 关系代数查询综合
- 3.2. 关系演算
 - 元组关系演算
 - 元组关系演算

4. 参考文献

关系操作

- 查询语言 (query language)
 - 用户用来从数据库中请求获取信息的语言，分为如下两类
 - 过程化语言 (procedural language): 用户指导系统执行一系列操作以计算所需结果。
 - 非过程语言 (nonprocedural language): 用户只需描述所需信息，而不用给出获取信息的具体过程。
- 关系代数 (Relational algebra) 是一种过程化查询语言，包括一个运算集合，这些运算以一个或两个关系为输入，产生一个新的关系作为结果。
- 多个关系运算组合成一个关系代数表达式 (relational-algebra expression) 其他运算可由基本运算定义，不增加关系代数表达能力
- 关系操作的特点是集合操作方式，即操作的对象和结果都是集合。这种操作方式也称为一次一个集合的方式。相应地，非关系数据模型的数据操作方式则为一次一个记录的方式。
- 关系模型中的关系操作有查询操作和更新操作(插入、删除和修改) 两大类。
- 查询操作是关系操作中最主要的部分。查询操作又可以分为选择 (select)、投影 (project)、连接 (join)、除 (divide)、并 (union)、交 (intersection)、差 (except)、笛卡尔积等。

关系操作可用两种方式来表示——代数方式和逻辑方式

- 关系代数是用**代数方式表达**的关系查询语言。
- 关系演算是用 textcolorpurple 逻辑方式表达的关系查询语言。
 - 关系演算又可按谓词变元的基本对象
 - 是元组变量还是域变量分为**元组关系演算**和 textcolorpurple 域关系演算。
- 对于**关系代数、元组关系演算和域关系演算**均是**抽象的查询语言**，在表达能力上是完全等价的。

1. 回顾

2. 关系模型

- 2.1. 关系数据结构
- 2.3. 关系完整性约束
- 2.4. 关系操作

3. 关系运算

3.1. 关系代数

- 传统的集合运算
- 专门的关系运算
- 关系代数查询综合

3.2. 关系演算

- 元组关系演算
- 域关系演算

4. 参考文献

关系运算

关系运算（Relational Operations）是数据库领域中用于操作关系数据的理论基础。关系代数（Relational Algebra）和关系演算（Relational Calculus）是关系运算的两大主要分支。它们提供了不同的方法来表达和实现对关系数据的查询和操作。

- **关系代数**: 是一种**过程化的查询语言**，通过一系列的操作符来逐步构建查询结果。它定义了一系列的基本操作，这些操作可以组合起来完成复杂的查询。
- **关系演算是基于逻辑的查询语言**，分为元组关系演算（Tuple Relational Calculus）和域关系演算（Domain Relational Calculus）。它是一种**非过程化的查询语言**，用户只需要描述所需的结果，而不需要指定如何获取这些结果。
 - 元组关系演算以**元组**为变量，通过**逻辑表达式**来描述查询结果。
 - 域关系演算以**属性值**为变量，通过**逻辑表达式**来描述查询结果。

1. 回顾

2. 关系模型

- 2.1. 关系数据结构
- 2.3. 关系完整性约束
- 2.4. 关系操作

3. 关系运算

3.1. 关系代数

- 传统的集合运算
- 专门的关系运算
- 关系代数查询综合

3.2. 关系演算

- 元组关系演算
- 域关系演算

4. 参考文献

- 关系模型由关系数据结构、关系操作与关系完整性约束3部分组成。
- 关系操作采用集合操作的方式，即操作的对象和结果都是集合，通常用代数的方式表示操作过程，称为关系代数。
- 关系代数是关系模型最重要的数据操作语言，它通过对关系的代数运算来表达查询，是一种抽象的查询语言。关系数据库操作的数学基础，是一种过程性查询语言。
- 关系代数是通过关系代数运算构成的表达式来表达查询。其运算是以一个或两个关系作为输入（即运算对象）产生一个新的关系作为结果。
- 逻辑运算符与比较运算符用于配合专门的关系运算符构造表达式。
- 基本的关系代数运算有选择、投影、集合并、集合差、笛卡尔积和更名等。

关系代数常用运算符

运算符	符号	含义
集合运算符	\cup	并
	$-$	差
	\cap	交
	\times	(广义) 笛卡尔积
专门的关系运算符	σ	选择
	Π	投影
	\bowtie	连接
	\div	除
逻辑运算符	\neg	非
	\wedge	与
	\vee	或

Table 4: 关系代数常用运算符及其含义

传统的集合运算

对两个关系进行的传统集合运算是二元运算，但并不是任意两个关系都可以进行集合运算。

- 除广义笛卡尔积运算外，其他的集合运算要求参加运算的关系必须符合相容关系。
- 其相容关系是指两个关系具有相同的列数，且两个关系各自的第 i 个属性来自同一个域。

前提假设

关系 r 和关系 s 具有相同的 n 个属性，且相应的属性取自同一个域，即两个关系的模式或结构相同。 t 是元组变量， $t \in r$ 表示 t 是 r 的一个元组。

(a) r

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_2	b_2	c_1

(b) s

A	B	C
a_1	b_2	c_2
a_1	b_3	c_2
a_2	b_2	c_1

并相容性

- 参与运算的两个关系及其相关属性之间有一定的对应性
- 定义：关系 R 与关系 S 存在相容性，当且仅当：
 - 关系 R 和关系 S 的属性数目必须相同；
 - 对于任意 i，关系 R 的第 i 个属性的域必须和关系 S 的第 i 个属性的域相同

假设： $R(A_1, A_2, \dots, A_n), S(B_1, B_2, \dots, B_m)$ R 和 S 满足并相容性： $n = m$ 并且
 $Domain(A_i) = Domain(B_i)$

并相容性的示例

STUDENT(SID char(10), Sname char(8), Age integer(3))

PROFESSOR(PID char(10), Pname char(8), Age integer(3))

关系 STUDENT 与关系 PROFESSOR 是相容的，因为：

- 关系 R 和关系 S 的属性数目都是 3
- 关系 R 的属性 SID 与关系 S 的属性 PID 的域都是 char(10)
- 关系 R 的属性 Sname 与关系 S 的属性 Sname 的域都是 char(8)
- 关系 R 的属性 Age 与关系 S 的属性 Age 的域都是 integer(3)

传统的集合运算: 并运算

关系 r 与关系 s 的并 (Union), 记作:

$$r \cup s = \{t | t \in r \vee t \in s\} \quad (3)$$

其结果关系仍为 n 目关系, 由属于 r 或属于 s 的所有元组组成。

Example (例 2-1)

对表5a与表5b所示的关系 r 与 s 做 $r \cup s$ 运算, 结果如表6所示。

Table 6: $r \cup s$ 结果

A	B	C
a_1	b_1	c_1
a_1	b_2	c_2
a_1	b_3	c_2
a_2	b_2	c_1

Tips:

通过并运算可以实现记录的添加与插入

传统的集合运算: 差

关系 r 与关系 s 的差 (Difference), 记作:

$$r - s = \{t | t \in r \wedge t \notin s\} \quad (4)$$

其结果关系仍为 n 目关系, 由属于 r 而不属于 s 的所有元组组成。

Example (例 2-2)

对表5a与表5b所示的关系 r 与 s 做 $r - s$ 运算, 结果如表7所示。

Table 7: $r - s$ 结果

A	B	C
a_1	b_1	c_1

Tips:

通过差运算可以实现记录的删除。

传统的集合运算: 交

关系 r 与关系 s 的交 (Intersection), 记作:

$$r \cap s = \{t | t \in r \wedge t \in s\} \quad (5)$$

其结果关系仍为 n 目关系, 由既属于 r 又属于 s 的所有元组组成。

Example (例 2-3)

对表5a与表5b所示的关系 r 与 s 做 $r \cup s$ 运算, 结果如表8所示。

Table 8: $r \cap s$ 结果

A	B	C
a_1	b_2	c_2
a_2	b_2	c_1

Tips:

- 如果两个关系没有相同的元组, 那么它们的交运算结果为空。
- 两个关系的并、差运算为基本运算, 而交运算为非基本运算, 它可以用差运算表示为: $r \cap s = r - (r - s)$ 。

传统的集合运算:(广义) 笛卡尔积 (Extended Cartesian Product)

- 两个分别为 n 目和 m 目的关系 r 和 s 的广义笛卡尔积是一个 $n + m$ 目元组的集合。
- 元组的前 n 列是关系 r 的一个元组，后 m 列是关系 s 的一个元组
- 若关系 r 有 k_r 个元组，关系 s 有 k_s 个元组，则关系 r 和 s 的笛卡尔积有 $k_r \times k_s$ 个元组。记作：

$$r \times s = \{t_r \dots t_s | t_r \in r \wedge t_s \in s\} \quad (6)$$

当 r 和 s 中有重名属性 A 时，则采用 $r.A$ 和 $s.A$ 分别命名对应的属性列。

Example (例 2-4)

对表5a与表5b所示的关系 r 与 s 做 $r \times s$ 运算，结果如表9所示。

Table 9: $r \times s$ 结果

$r.A$	$r.B$	$r.C$	$s.A$	$s.B$	$s.C$
a_1	b_1	c_1	a_1	b_2	c_2
a_1	b_1	c_1	a_1	b_3	c_2
a_1	b_1	c_1	a_2	b_2	c_1
a_1	b_2	c_2	a_1	b_2	c_2
a_1	b_2	c_2	a_1	b_3	c_2
a_1	b_2	c_2	a_2	b_2	c_1
a_2	b_2	c_1	a_1	b_2	c_2
a_2	b_2	c_1	a_1	b_3	c_2
a_2	b_2	c_1	a_2	b_2	c_1

成绩管理数据库 ScoreDB 的实例数据

对于数据库 ScoreDB, Class 关系, Course 关系。

(a) Class 关系

calssNo	className	institute	grade	classNum
AC2103	会计学 21(3) 班	会计学院	2021	46
CS2101	计算机 21(1) 班	信息学院	2021	48
IS2202	信息系统 22(2) 班	信息学院	2022	43

(b) Course 关系

courseNo	courseName	creditHour	courseHour	priorCourse
AC001	基础会计	3	48	NULL
CN028	大学语文	3	48	NULL
CS012	操作系统	5	80	NULL
CS015	数据库系统	4	64	CS012

成绩管理数据库 ScoreDB 的实例数据

Example (例 2-5)

对于数据库 ScoreDB, $Class \times Course$ 的关系为笛卡尔积。

$Class \times Course$ 的关系为笛卡尔积, 如表11所示。

Table 11: 笛卡尔积 $Class \times Course$ 的结果关系

classNo	className	institute	grade	classNum	courseNo	courseName	creditHour	courseHour	priorCour
AC2103	会计学 21(3) 班	会计学院	2021	46	AC001	基础会计	3	48	NULL
AC2103	会计学 21(3) 班	会计学院	2021	46	CN028	大学语文	3	48	NULL
AC2103	会计学 21(3) 班	会计学院	2021	46	CS012	操作系统	5	80	NULL
AC2103	会计学 21(3) 班	会计学院	2021	46	CS015	数据库系统	4	64	CS012
AC2101	计算机 21(1) 班	信息学院	2021	48	AC001	基础会计	3	48	NULL
AC2101	计算机 21(1) 班	信息学院	2021	48	CN028	大学语文	3	48	NULL
AC2101	计算机 21(1) 班	信息学院	2021	48	CS012	操作系统	5	80	NULL
AC2101	计算机 21(1) 班	信息学院	2021	48	CS015	数据库系统	4	64	CS012
IS2202	信息系统 22(2) 班	信息学院	2022	43	AC001	基础会计	3	48	NULL
IS2202	信息系统 22(2) 班	信息学院	2022	43	CN028	大学语文	3	48	NULL
IS2202	信息系统 22(2) 班	信息学院	2022	43	CS012	操作系统	5	80	NULL
IS2202	信息系统 22(2) 班	信息学院	2022	43	CS015	数据库系统	4	64	CS012

专门的关系运算：选择

选择 (*Selection*) 操作是在关系 r 中查找满足给定谓词（即选择条件）的所有元组，记作：

$$\sigma_P(r) = \{t | t \in r \wedge P(t)\} \quad (7)$$

其中：

- “ σ ” 为选择运算符；
- P 表示谓词（即选择条件），
 - 由运算对象、比较运算符和逻辑运算符组成逻辑表达式，
 - 是一个逻辑表达式，取值为“真”或“假”。
- 实际上，选择运算就是从关系 r 中选取使逻辑表达式 P 为真的元组的过程，是 **从行的角度进行的运算**。

专门的关系运算: 选择举例

Example (例 2-6)

在数据库 *ScoreDB* 中,

- 查找 2021 级的所有班级情况。
 - 查找所有 2005 年及以后出生的女学生情况。
-
- $\sigma_{grade=2021}(Class)$, 结果如表12所示。
 - $\sigma_{year(birthday)>=2021 \wedge sex='女'}(Student)$, 结果如表13所示。

Table 12: 查找 2021 级的所有班级结果

classNo	className	institute	grade	classNum
AC2103	会计学 21(3) 班	会计学院	2021	46
CS2101	计算机 21(1) 班	信息学院	2021	48

Table 13: 查找所有 2021 年以后出生的女学生结果

studentNo	studentName	sex	birthday	native	nation	classNo
2103010	李宏冰	女	2005-03-09	太原	蒙古族	AC2103

专门的关系运算：投影

投影运算是单目运算，按照指定的顺序从左至右依次取出关系 r 中的若干条属性列，删去重复元组，组成一个新关系。关系 r 上的投影 (Projection) 是从 r 中选择出若干属性列组成新的关系。记作：

$$\Pi_A(r) = \{t[A] | t \in r\} \quad (8)$$

其中， A 为关系 r 的属性集合。

Tips:

选择运算与投影运算的区别在于，

- 选择运算从关系的水平方向进行运算，
- 而投影运算从关系的垂直方向进行运算。

专门的关系运算：投影举例

Example (例 2-8)

在数据库 *ScoreDB* 中，查找所有学生的姓名和民族。

$$\Pi_{studentName, nation}(Student)$$

结果如表14所示。

Table 14: 查找 2021 级的所有班级结果

studentName	nation
李小勇	汉族
王红	汉族
李宏冰	蒙古族
刘方晨	傣族
王红敏	蒙古族

Example (例 2-9)

在数据库 *ScoreDB* 中，查找所有“蒙古族”学生的姓名和籍贯。

$$\Pi_{studentName, native}(\sigma_{nation='蒙古族'}(Student))$$

结果如表15所示。

Table 15: 查找所有“蒙古族”学生的姓名和籍贯结果

studentName	native
李宏冰	太原
王红敏	上海

专门的关系运算: 连接 (natural join)

- 连接 (Join)也称为 θ 连接。记为 $A \text{ op } B$, 其中 A, B 分别为关系 r 和 s 中的属性的数量相等且可比的连接属性集, op 为比较运算符。 θ 连接是从两个关系的笛卡尔积中选取连接属性间满足谓词 θ 的所有元组。记作:

$$r \bowtie_{\theta} s = \{t_r \cdot t_s | t_r \in r \wedge t_s \in s \wedge (r.A \text{ op } s.B)\} \quad (9)$$

- θ 连接运算是从关系 r 和 s 的笛卡尔积 $r \times s$ 中, 选取 r 关系在 A 属性集上的值与 s 关系在 B 属性集上的值满足连接谓词 θ 的所有元组, 即

$$r \bowtie_{\theta} s = \sigma_{\theta}(r \times s) \quad (10)$$

其中, 式 (10) 首先形成两个参数关系的笛卡尔积, 然后基于两个关系模式中都出现的属性上的相等性进行选择 (只考虑两个关系在所有相同属性有相同值的元组组成的元组对), 最后去除重复属性 (并不重复记录在两个关系模式中都出现的属性)。

专门的关系运算: 连接

- 连接运算中有两种最常用、最重要的连接，一种是等值连接 (equijoin)，另一种是自然连接 (natural join)。 θ 为等值比较谓词的连接运算称为等值连接。
- 自然连接是一种特殊的等值连接，它要求两个参与连接的关系具有公共的属性集，并在这个公共属性集上进行等值连接；同时，还要求将连接结果中的重复属性列去除掉，即在公共属性集中的列只保留一次。
- 形式化定义：设 $r(R)$ 和 $s(S)$ （模式分别为 R 和 S 的两个关系），其自然连接的结果 $r \bowtie s$ 是模式 $R \cup S$ 上的一个关系。

$$r \bowtie s = \Pi_{R \cup S}(\sigma_{r.A_1=s.A_1 \wedge r.A_2=s.A_2 \wedge \dots \wedge r.A_n=s.A_n}(r \times s)) \quad (11)$$

其中 $R \cap S = \{A_1, A_2, \dots, A_n\}$ 。

- 如果 r 和 s 不含有任何相同属性，即 $R \cap S = \emptyset$ ，则 $r \bowtie s = r \times s$ 。
- 所列出的属性的顺序：两个关系模式的相同属性排在最前，只属于第一个关系模式的属性其次，只属于第二个关系模式的属性最后。

专门的关系运算：连接举例

Example (例 2-10)

在数据库 *ScoreDB* 中，查找所有 2022 级的“蒙古族”学生的姓名

分析：

1. $\sigma_{nation='蒙古族'}(Student)$ 可以找到所有蒙古族学生的情况，但关系 *Student* 中没有年级的信息，因此必须将关系 *Student* 与关系 *Class* 关联起来。
2. 根据模式导航图可知，关系 *Student* 与关系 *Class* 可通过外码 *classNo* 关联起来，这种外码引用关系可通过自然连接表示

$$Student \bowtie Class = \sigma_{Student.classNo=Class.classNo}(Student \times Class)$$

3. 最后的查询可表达为：

$$\Pi_{studentName}(\sigma_{nation='蒙古族'}(Student) \bowtie \sigma_{grade=2022}(Class))$$

专门的关系运算: 连接举例

$$\Pi_{studentName}(\sigma_{nation='蒙古族'}(Student) \bowtie \sigma_{grade=2022}(Class)) \quad (12)$$

$$= \Pi_{studentName}(\sigma_{Student.classNo=Class.classNo}(\sigma_{nation='蒙古族'}(Student) \times \sigma_{grade=2022}(Class))) \quad (13)$$

$$= \Pi_{studentName}(\sigma_{Student.classNo=Class.classNo}(\sigma_{nation='蒙古族' \wedge grade=2022}(Student \times Class))) \quad (14)$$

$$= \Pi_{studentName}(\sigma_{nation='蒙古族' \wedge grade=2022 \wedge Student.classNo=Class.classNo}(Student \times Class)) \quad (15)$$

$$= \Pi_{studentName}(\sigma_{nation='蒙古族' \wedge grade=2022}(\sigma_{Student.classNo=Class.classNo}(Student \times Class))) \quad (16)$$

$$= \Pi_{studentName}(\sigma_{nation='蒙古族' \wedge grade=2022}(Student \bowtie Class)) \quad (17)$$

专门的关系运算: 连接举例

在数据库 *ScoreDB* 中, 查找课程号为 *AC001* 课程的考试中比学号为 2103045 的学生考得更好的所有学生的姓名和成绩。

- 找出学号为 2103045 的学生在课程号为 *AC001* 的课程中的成绩元组 (结果关系记为 r_1), 可表达为: $(\sigma_{studentNo='2103045' \wedge courseNo='AC001'}(Score)) \text{ AS } r_1$
- 找出选修了课程号为 *AC001* 课程的所有学生的成绩元组 (结果关系记为 r_2), 其查询可表达为: $(\sigma_{courseNo='AC001'}(Score)) \text{ AS } r_2$
- 将关系 r_1 与关系 r_2 进行 θ 连接 (结果关系记为 r_3), 其查询可表达为:
 $r_1 \bowtie_{r_1.score < r_2.score} r_2 = \sigma_{r_1.score < r_2.score}(r_1 \times r_2)$
- 将关系 r_3 与学生关系 *Student* 按外码 *studentNo* 进行自然连接, 并对连接结果在属性 *studentName* 和 $r_2.score$ 上进行投影, 其查询可表达为:

$$\begin{aligned}& \Pi_{studentName, r_2.score}(\Pi_{r_2.studentNo, r_2.score}(r_3) \bowtie Student) \\&= \Pi_{studentName, r_2.score}(\sigma_{r_2.studentNo=Student.studentNo}(\Pi_{r_2.studentNo, r_2.score}(r_3) \times Student)) \\&= \Pi_{studentName, r_2.score}(\sigma_{r_2.studentNo=Student.studentNo}(\Pi_{r_2.studentNo, r_2.score}(\\&\quad \sigma_{r_1.score < r_2.score}(r_1 \times r_2)) \times Student))\end{aligned}$$

专门的关系运算: 连接举例

studentNo	courseNo	term	score
2103045	AC001	21221	52
2103045	AC001	22231	94

(a) 关系 $r1$

studentNo	courseNo	term	score
2101008	AC001	21221	76
2103010	AC001	21221	92
2103045	AC001	21221	52
2103045	AC001	22231	94
2202002	AC001	22231	98
2202005	AC001	23241	88

(b) 关系 $r2$

studentName	score
王红	76
李宏冰	92
王红	94
刘方晨	98
王红敏	88

(d) 最后结果

$r1.studentNo$	$r1.courseNo$	$r1.term$	$r1.score$	$r2.studentNo$	$r2.courseNo$	$r2.term$	$r2.score$
2103045	AC001	21221	52	2101008	AC001	21221	76
2103045	AC001	21221	52	2103010	AC001	21221	92
2103045	AC001	21221	52	2103045	AC001	22231	94
2103045	AC001	21221	52	2202002	AC001	22231	98
2103045	AC001	21221	52	2202005	AC001	23241	88
2103045	AC001	22231	94	2202002	AC001	22231	98

(c) 关系 $r3$

Figure 6: θ 连接的计算过程

专门的关系运算：连接 $r \bowtie_{B \leq C} s$

r	
A	B
α	1
β	2

s	
C	D
1	x
1	y
3	z

$r \times s$				
A	B	C	D	
α	1	1	x	
α	1	1	y	
α	1	3	z	
β	2	1	x	
β	2	1	y	
β	2	3	z	

$r \bowtie_{B \leq C} s r$				
A	B	C	D	
α	1	1	x	
α	1	1	y	
α	1	3	z	
β	2	3	z	

专门的关系运算: 连接

- 外连接 (outer-join) 运算: 连接运算的扩展, 可处理缺失的信息 (因为不满足自然连接条件而不能在自然连接结果中找到的元组), 避免信息的丢失。
- 外连接运算与自然连接运算类似, 不同之处在于外连接在连接计算结果中添加额外的带空值的元组, 以此保留在连接中丢失的元组。

外连接运算有三种形式:

- 左外连接 (left outer join) (\bowtie): 取出左侧关系中所有与右侧关系的任一元组都不匹配的元组, 用空值填充所有来自右侧关系的属性, 再把产生的元组加到自然连接的结果中。所有来自左侧关系的信息在左外连接结果中都得到保留。
- 右外连接 (right outer join) (\bowtie): 与左外连接相对称, 用空值填充来自右侧关系的所有与左侧关系任一元组都不匹配的元组, 将结果加到自然连接的结果中。所有来自右侧关系的信息在右外连接中都得到保留。
- 全外链接 (full outer join) (\bowtie): 既做左外连接又做右外连接, 既填充左侧关系中与右侧关系的任一元组都不匹配的元组, 又填充右侧关系中与左侧关系任一元组都不匹配的元组, 并把结果都加到连接的结果中。

专门的关系运算: 连接

R		
A	B	C
a1	b1	5
a1	b2	6
a2	b3	8
a2	b4	12

S	
B	E
b1	3
b2	7
b3	10
b3	2
b5	2

R \times S			
A	B	C	E
a1	b1	5	3
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2
a2	b4	12	NULL

R \times S			
A	B	C	E
a1	b1	5	3
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2
NULL	b5	NULL	2

R \bowtie S			
A	B	C	E
a1	b1	5	3
a1	b2	6	7
a2	b3	8	10
a2	b3	8	2
a2	b4	12	NULL
NULL	b5	NULL	2

专门的关系运算: 除运算

为了叙述方便, 先给出象集的概念。

Definition (象集)

给定一个关系 $r(X, Y), X$ 和 Y 为属性集。

$\forall t \in r$, 记 $t[X] = x$, 则在关系 r 中属性集 X 的某个取值 x, x 在 r 中的象集(Image Set) 是指在关系 r 中所有与 x 配对的 Y 值的集合, 即: 对于 $x \in \Pi_x(r), x$ 的象集记为 Y_x :

$$Y_x = \{t[Y] | t \in r, t[X] = x\} \quad (18)$$

式(18)表示关系 r 中属性集 X 上取值为 x 的所有元组在属性集 Y 上的投影。

- $t[Y]$ 表示元组 t 在属性组 Y 上的取值
- $t[X] = x$ 表示元组 t 在属性组 X 上的取值等于 x

在数据库 ScoreDB 中, 关系 Score, 设 $X = \{studentNo\}, Y = \{courseNo, term, score\}$, 则在关系 Score 中值'2101008' 和'2202005' 的象集 $Y_{2101008'}$ 和 $Y_{2202005'}$ 的结果:

courseNo	term	score
AC001	21221	76
CN028	21221	86
CS012	21222	93
CS015	22231	96

(a) 象集 $B_{2101008'}$

courseNo	term	score
AC001	23241	88
CS012	22232	90
CS015	23241	87

(b) 象集 $B_{2202005'}$

专门的关系运算：除运算

- 除 (Division) 运算是二目运算,
- 设有关系 $r(X, Y)$ 与关系 $s(Y, Z)$, 其中 X, Y, Z 为属性或属性集。 r 中的 Y 和 s 中的 Y 可以有不同的属性名, 但必须取自相同的域。 r 与 s 的除运算得到一个新的关系 $P(X)$, P 是 r 中满足下列条件的元组在属性 X 上的投影: 元组在 X 上的分量值 x 的象集 Y_x 包含 s 在 Y 上投影的集合。
- 记作:

$$r \div s = \{t_r[X] | t_r \in r \wedge \prod_y(s) \subseteq Y_x\} \quad (19)$$

其中, Y_x 为 x 在 r 中的象集, x 为 $t_r[X]$ 。

象集实际上是一次选择运算加上一次投影运算。如, 当 x 为 $t_r[X]$ 时, x 的象集可理解为先在所有元组中选择属性 X 中分量值为 x 的行, 再投影出不含属性 X 的所有列。

专门的关系运算: 除运算

- 设关系 $r(R)$ 除以 $s(S)$ 的结果为 $t(T)$ ，则 $t(T)$ 包含所有在 $r(R)$ 但不在 $s(S)$ 中的属性及其值，且 $t(T)$ 的元组与 $s(S)$ 的元组的所有组合都在 $r(R)$ 中。
- 设关系 $r(R)$ 和 $s(S)$ ，属性集 S 是 R 的子集，即 $S \subseteq R$ ，则关系 $r \div s$ 是关系 r 中满足下列条件的元组在属性集 $R - S$ 上的投影： $\forall t_r \in r$ ，记 $x = t_r[R - S]$ ，则关系 r 中属性集 $R - S$ 的取值 x 的象集 S_x 包含关系 s 。记作

$$r \div s = \{t_r[R - S] \mid t_r \in r \wedge s \subseteq S_x\} \quad (20)$$

不含属性 X 的所有列。

专门的关系运算: 除运算

设 R 和 S 是两个关系 $X = r(R) = \{A_1, \dots, A_m, B_1, \dots, B_n\}$, $Y = r(S) = \{B_1, \dots, B_n\}$, $Z = X - Y = \{A_1, \dots, A_m\}$, $R \div S$ 的属性为 $\{A_1, \dots, A_m\}$

$$R \div S = \{t | t \in \Pi_Z(R) \wedge \forall u \in S (t_u \in R)\} \quad (21)$$

t_u 是指元组 t 与元组 u 相连接构成的新元组

$$R \div S = \Pi_Z(R) - \Pi_Z((\Pi_Z(R) \times S) - R) \quad (22)$$

专门的关系运算: 除运算举例

Example (例 2-12)

关系 r 与关系 s 做 $r \div s$ 运算, 运算过程如图所示。

关系 r		
A	B	C
a	b	c
d	b	c
a	e	c
a	e	f
d	b	f
a	e	g
a	e	h
a	b	l

关系 s	
C	
c	
f	
g	
h	

1. $\Pi_{r-s}(r)$	
A	B
a	b
d	b
a	b
d	b
a	e

2. $\Pi_{r-s}(r) \times s$		
A	B	C
a	b	c
d	b	c
a	e	c
a	b	f
d	b	f
a	e	f
a	b	g
d	b	g
a	e	g
a	b	h
d	b	h
a	e	h

3. $\Pi_{r-s}(r) \times s - r$		
A	B	C
a	b	f
a	b	g
d	b	g
a	b	h
d	b	h

专门的关系运算：除运算举例

4. $\Pi_{r-s}(\Pi_{r-s}(r) \times s - r)$	
A	B
a	b
d	b

5. $\Pi_{r-s}(r) - \Pi_{r-s}(\Pi_{r-s}(r) \times s - r)$	
A	B
a	e

举例：除运算

Example (例 2-13)

需要查找修读过信息学院开设的所有课程的学生学号，如何表达查询？

- 查找出修读过信息学院课程的所有学生：

$$r_1 = \Pi_{studentNo, courseNo}(\sigma_{courseNo \text{ LIKE } 'CS%' \text{ AND } studentNo \in r_2}(Score))$$

- 找出信息学院开设的所有课程： $r_2 = \Pi_{courseNo}(\sigma_{courseNo \text{ LIKE } 'CS%' \text{ AND } courseNo \in r_1}(Course))$
- 比较图7(a) 和 (b)，修读过信息学院开设的所有课程的学生就是关系 r_1 中满足“ $courseNo$ 列包含关系 r_2 的所有行”的那些学生。

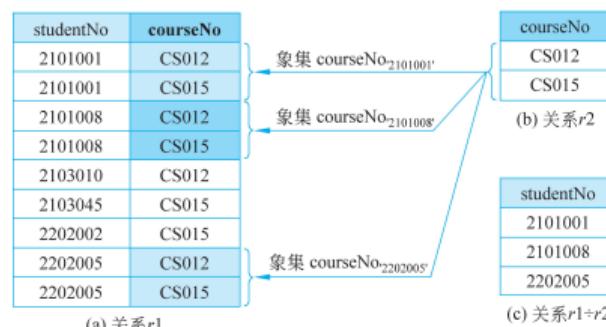


Figure 7: 除运算实例

举例：除运算

- 关系数据库 *ScoreDB* 中，求以下结果：

$$\Pi_{studentNo, courseNo}(Score) \div (\Pi_{courseNo}(\sigma_{courseName='基础会计' \vee courseName='操作系统'}(Course)))$$

表示查找既选修了“基础会计”又选修了“操作系统”课程的学生学号，查询结果如图8(c) 所示。

其中，被除数 $\Pi_{studentNo, courseNo}(Score)$ 和除数

$\Pi_{courseNo}(\sigma_{courseName='基础会计' \vee courseName='操作系统'}(Course))$ 的查询结果分别如图8(a) 和 (b) 所示。

- 关系数据库 *ScoreDB* 中，求以下结果：

$$\Pi_{studentNo, studentName}(((\Pi_{studentNo, courseNo}(Score)) \div (\Pi_{courseNo}(Course))) \bowtie Student)$$

表示查找选修了所有课程的学生的学号和姓名，查询结果如图8(e) 所示。

其中除数 $\Pi_{courseNo}(Course)$ 的查询结果如图8(d) 所示。

举例：除运算

关系数据库 *ScoreDB* 中，求以下结果：

$$\Pi_{courseNo, courseName}(((\Pi_{courseNo, studentNo}(Score)) \div (\Pi_{studentNo}(\sigma_{sex='男'}(Student)))) \\ \bowtie Course)$$

表示查找被所有男同学选修过的课程的课程号和课程名，查询结果如图8(g) 所示。其中除数 $\Pi_{studentNo}(\sigma_{sex='男'}(Student))$ 的查询结果如图8(f) 所示。关系数据库 *ScoreDB* 中，求以下结果：

$$\Pi_{studentNo, studentName}(((\Pi_{studentNo, courseNo}(Score)) \div ((\Pi_{courseNo, studentNo}(Score)) \\ \div (\Pi_{studentNo}(\sigma_{sex='男'}(Student))))) \bowtie (\sigma_{sex='女'}(Student)))$$

表示查找选修了被所有男同学同时选修过的所有课程的女学生的学号和姓名，查询结果如图8(h) 所示。

举例：除运算

studentNo	courseNo
2101001	CN028
2101001	CS012
2101001	CS015
2101008	AC001
2101008	CN028
2101008	CS012
2101008	CS015
2103010	AC001
2103010	CN028
2103010	CS012
2103045	AC001
2103045	CN028
2103045	CS015
2202002	AC001
2202002	CN028
2202002	CS015
2202005	AC001
2202005	CS012
2202005	CS015

(a) $\Pi_{studentNo, courseNo}(Score)$ 的结果

courseNo
AC001
CS012

(b) $\Pi_{courseNo.}(\sigma_{courseName='基础会计' \vee courseName='操作系统'}(Course))$ 的结果

studentNo
2101008
2103010
2202005

(c) 查询(1)的结果

courseNo
AC001
CN028
CS012
CS015

(d) $\Pi_{courseNo}(Course)$ 的结果

studentNo	studentName
2101008	王红

(e) 查询(2)的结果

courseNo	courseName
CN028	大学语文
CS015	数据库系统

(g) 查询(3)的结果

studentNo
2101001
2101008
2103045

(f) $\Pi_{studentNo}(\sigma_{sex='男'}(Student))$ 的结果

studentNo	studentName
2202002	刘方晨

(h) 查询(4)的结果

Figure 8: 除运算实例结果

关系代数查询综合举例

给定一个查询需求，构造其关系代数表达式的步骤：

- 明确该查询涉及到哪些属性；
- 明确该查询涉及到哪些关系；
- 根据数据库模式导航图，通过多对一联系（或一对多联系）把所有涉及的关系连接起来，每一个多对一联系（或一对多联系）都可以表示为外码属性的自然连接。

关系代数查询综合举例

Example (例 2-14)

查找“蒙古族”学生所修各门课程的情况，要求输出学生姓名、课程名和成绩。

- 该查询共涉及 4 个属性，分别是民族 nation、姓名 studentName、课程名 courseName 和成绩 score，其中，nation 属性用于选择条件 notion='蒙古族'。
- 共涉及 3 个关系，分别是学生关系 Student、课程关系 Course 和成绩关系 Score。
- 成绩关系 Score 通过外码 studentNo、courseNo 分别与学生关系 Student、课程关系 Course 建立多对一的联系。

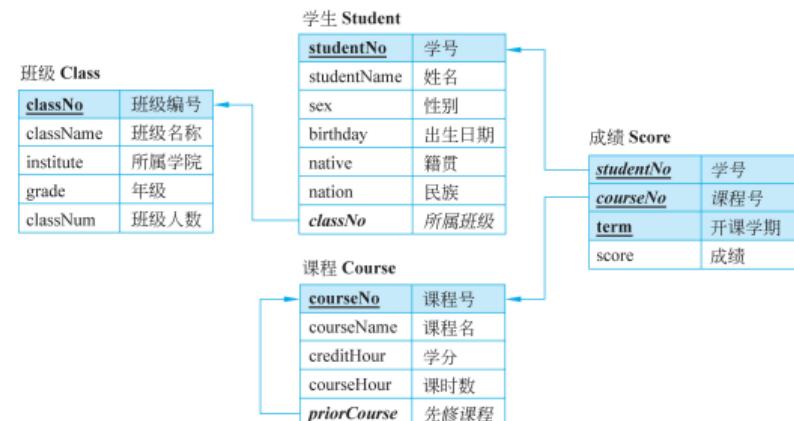


Figure 9: 成绩管理数据库 ScoreDB 的模式导航图

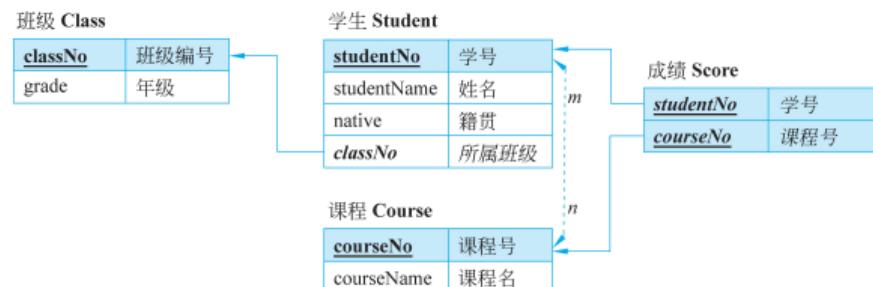
$$\Pi_{studentName, courseName, score} (\sigma_{nation='蒙古族'} ((Student \bowtie Score) \bowtie Course)) \quad (23)$$

关系代数查询综合举例

Example (例 2-15)

查找 2021 级的“南昌”籍同学修读了哪些课程，要求输出学生姓名、课程名。

- 该查询共涉及 4 个属性，分别是年级 grade、籍贯 native、姓名 studentName 和课程名 courseName，其中年级 grade 和籍贯 native 用于选择条件。
- 共涉及 3 个关系，分别是班级关系 Class、学生关系 Student 和课程关系 Course。
- 学生关系 Student 与班级关系 Class 之间是多对一联系；学生关系 Student 和课程关系 Course 之间是多对多联系，并借助成绩关系 Score 才能建立。因此，该查询还需涉及 Score 关系。



$$\Pi_{studentName, courseName}(((\sigma_{grade=2021}(Class) \bowtie \sigma_{native='南昌'}(Student)) \bowtie Score) \bowtie Course)$$

关系代数查询综合举例

Example (例 2-16)

SCDB 数据库中，查找“吴文君”老师教过的 2022 级学生的姓名。

- 该查询共涉及 3 个属性，分别是职工名 teacherName、年级 grade 和学生姓名 studentName，职工名 teacherName 和年级 grade 都是用于选择条件。
- 共涉及 3 个关系，分别是教师关系 Teacher、班级关系 Class 和学生关系 Student。

$$\Pi_{studentName}(((\sigma_{grade=2016}(Class) \bowtie Student) \bowtie SC) \bowtie CourseClass) \bowtie \sigma_{teacherName='吴文君'}(Teacher)$$

- 学生关系 Student 与班级关系 Class 之间是多对一联系；学生关系 Student 和教师关系 Teacher 之间是多对多联系，这种多对多联系不是直接通过一个联系关系就能建立。



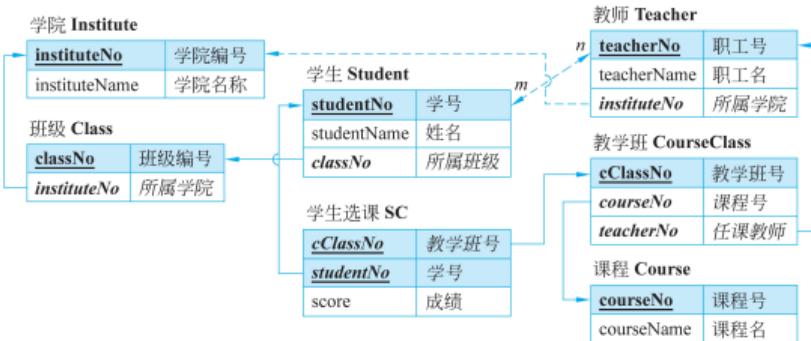
关系代数查询综合举例

Example (例 2-17)

查找“吴文君”老师在“操作系统”课程中教过的“信息学院”学生的姓名。

- 该查询共涉及 4 个属性，分别是职工名 teacherName、课程名 courseName、学院名称 instituteName 和学生姓名 studentName，职工名 teacherName、课程名 courseName 和学院名称 instituteName 都是用于选择条件。
- 共 4 个关系，分别是教师关系 Teacher、课程关系 Course、学院关系 Institute 和学生关系 Student。
- 学生关系 Student 与班级关系 Class 之间是多对一联系。班级关系 Class 又与学院关系 Institute 之间是多对一联系。学生关系 Student 和教师关系 Teacher 之间是多对多联系，这种多对多联系不是直接通过一个联系关系就能建立。

关系代数查询综合举例

$$\begin{aligned} \Pi_{studentName}(((\Pi_{classNo}(\sigma_{instituteName='信息学院'}(Institute) \bowtie Class) \bowtie Student) \bowtie SC) \\ \bowtie (CourseClass \bowtie \sigma_{courseName='操作系统'}(Course))) \\ \bowtie \sigma_{teacherName='吴文君'}(Teacher)) \end{aligned}$$


说明：由于关系 *Institute* 与关系 *Teacher* 可以通过外码 *InstituteNo* 进行自然连接，因此，在 $\Pi_{classNo}(\sigma_{instituteName='信息学院'}(Institute) \bowtie Class)$ 中使用投影操作的目的是去掉无用属性（主要是为了去掉关系 *Institute* 的主码 *InstituteNo*），以避免产生歧义。

关系代数查询综合举例

以订货系统为例，运用关系代数相关知识，写出实际查询操作的关系代数表达式。现给出各关系如表16所示。

(a) Customer

CustomerNo	CustomerName	Adds	Tel.
0145	沈风	城南	03911084145
0132	韩雅	城北	03212084134
0268	杨颂	城西	05813084555

(b) Product

ProductNo	ProductName	Pdata	Price
0114	蛋挞	2019-1-1	5
0117	椰蓉球	2019-1-1	10
0275	羊角包	2019-1-2	20
0213	三明治	2019-1-2	15
0235	鲜花饼	2019-1-3	20
0316	鲜牛奶	2019-1-3	10
0384	吐司面包	2019-1-3	10
0347	风味酸奶	2019-1-4	5

(c) ProductCount

CustomerNo	ProductNo	Num
0145	0117	20
0145	0213	10
0132	0114	15
0132	0275	10
0132	0347	20
0268	0117	20
0268	0384	5
0268	0316	10

Table 16: ProductDB

关系代数查询综合举例

1. 查询客户“沈风”的联系电话。

$$\Pi_{Tel.}(\sigma_{CustomerName='沈风'}(Customer)) \quad (24)$$

2. 查询商品“鲜牛奶”的订购信息。

$$\Pi_{ProductNo}(\sigma_{ProductName='鲜牛奶'}(Product)) \bowtie ProductCount \quad (25)$$

3. 查询客户“韩雅”订购的商品信息。

$$\Pi_{ProductNo}(\Pi_{CustomerNo}(\sigma_{CustomerName='韩雅'}(Customer)) \bowtie ProductCount) \bowtie Product \quad (26)$$

关系代数查询综合举例

Example (例 2-19: 银行信息数据库实例)

银行关系: branch (BranchName, BranchCity, Assets)

客户关系: customer (CustomerName, CustomerStreet, CustomerCity)

账户: account (AccountNo, BranchName, Balance)

贷款: loan (LoanNo, BranchName, Amount)

存款人: depositor (CustomerName, AccountNo)

贷款人: borrower (CustomerName, loanNo)

- 查询所有额度超过 1200 的贷款信息
- 查询所有额度超过 1200 的贷款号
- 查询在银行中贷过款或存过款的客户名
- 查询所有在银行中贷过款的客户名、贷款号、贷款额
- 查询所有即在“Downtown”银行有存款账户又在“Uptown”银行有存款账户的所有客户名

关系代数查询综合举例

- 查询所有额度超过 2000 的贷款信息

$$\sigma_{Amount > 2000}(loan)$$

- 查询所有额度超过 3000 的贷款号

$$\Pi_{LoanNo}(\sigma_{Amount > 3000}(loan))$$

- 查询在银行中贷过款或存过款的客户名

$$\Pi_{CustomerName}(borrower) \cup \Pi_{CustomerName}(depositor)$$

- 查询所有在银行中贷过款的客户名、贷款号、贷款额

$$\Pi_{CustomerName, LoanNo, Amount}(borrowr \bowtie loan)$$

- 查询所有即在“Downtown”银行有存款账户又在“Uptown”银行有存款账户的所有客户名

$$\Pi_{CustomerName}(\sigma_{BranchName='Downtown'}(depositor \bowtie account)) \cap$$
$$\Pi_{CustomerName}(\sigma_{BranchName='Uptown'}(depositor \bowtie account))$$

关系代数（总结）

运算符	符号	含义
集合运算符	\cup	并
	$-$	差
	\cap	交
	\times	(广义) 笛卡尔积
专门的关系运算符	σ	选择
	Π	投影
	\bowtie	连接
	\div	除
逻辑运算符	\neg	非
	\wedge	与
	\vee	或

- 连接：连接、等值连接、自然连接
- 给定一个查询需求，构造其关系代数表达式的步骤：
 1. 明确该查询涉及到哪些属性；
 2. 明确该查询涉及到哪些关系；
 3. 根据数据库模式导航图，通过多对一联系（或一对多联系）把所有涉及的关系连接起来，每一个多对一联系（或一对多联系）都可以表示为外码属性的自然连接。

本章作业

对于图5所示的成绩管理数据库 ScoreDB 的模式导航图, 试写出如下查询的关系代数表达式, 并给出其查询结果。

- (1) 查找籍贯为“上海”的全体学生。
- (2) 查找 2005 年元旦以后出生的全体男同学。
- (3) 查找信息学院非汉族同学的学号、姓名、性别及民族。
- (4) 查找 2022—2023 学年第二学期 (22232) 开设课程的课程号、课程名和学分。
- (5) 查找选修了“操作系统”的学生学号、成绩及姓名。
- (6) 查找班级名称为“会计学 21(3) 班”的学生在 2021—2022 学年第一学期 (21221) 选课情况, 要求显示学生姓名、课程号、课程名和成绩。
- (7) 查找至少选修了一门其直接先修课程 (号) 为 CS012 的课程的学生学号和姓名。
- (8) 查找选修了 2022—2023 学年第一学期 (22231) 开设的全部课程的学生学号和姓名。
- (9) 查找至少选修了学号为 2103010 的学生所选课程的学生学号和姓名。

作业

Example

设有如图所示的关系 R, W 和 D, 计算:

- $R_1 = \Pi_{[2],[1],[6]}(\sigma_{[3]=[5]}(R \times D))$
- $R_2 = R \div D$

Table 17: R

P	Q	T	Y
2	b	c	d
9	a	e	f
2	b	e	f
9	a	d	e
7	g	e	f
7	g	c	d

Table 18: D

T	Y
c	d
e	f

作业

在图书管理数据库中，有如下三个关系：

- 图书信息关系： $B(BNo, BNAME, AUTHOR, TYPE)$ ，其中 BNo 为图书编号， $BNAME$ 为书名， $AUTHOR$ 为作者， $TYPE$ 为类别；
- 学生信息关系： $S(SNo, SNAME, CLASS)$ ，其中 SNo 为学号， $SNAME$ 为学生姓名， $CLASS$ 为班级号；
- 借阅信息关系： $L(SNo, BNo, DATE)$ ，其中 SNo 为借阅人学号， BNo 为被借阅图书编号， $DATE$ 为借阅日期。

使用关系代数回答以下问题：

- (1) 查询借阅了“《西游记》”这本书的学生的班级
- (2) 查询“201”班学生借阅图书的书名
- (3) 查询“小明”借过，但“小李”没有借过的图书的编号
- (4) 查询借阅过“《红楼梦》”这本书的总学生数

1. 回顾

2. 关系模型

- 2.1. 关系数据结构
- 2.3. 关系完整性约束
- 2.4. 关系操作

3. 关系运算

- 3.1. 关系代数
 - 传统的集合运算
 - 专门的关系运算
 - 关系代数查询综合

3.2. 关系演算

- 元组关系演算
- 域关系演算

4. 参考文献

元组关系演算

- 元组关系演算 (Tuple relational calculus)
 - 非过程化的查询语言
 - 只需描述所需信息，而不给出获得该信息的具体过程
- 元组关系演算表达式为

$$\{t | P(t)\} \quad (27)$$

- P 为一个公式，公式中可出现多个元组变量，如果元组变量不被 \exists 或 \forall 修饰则称为自由变量，否则称为受限变量。
- 公式由原子构成。原子可以为如下形式之一：
 - ▶ $s \in r$, 其中 s 为元组变量, r 为关系；
 - ▶ $s[x]\theta u[y]$, 其中 s, u 为元组变量, x 为 s 所基于的关系模式中的一个属性, y 为 u 所基于的关系模式中的一个属性, θ 为比较运算符 ($<, \leq, =, \neq, >, \geq$) (要求 x 和 y 所属域成员可用 θ 比较)；
 - ▶ $s[x]\theta c$, 其中 s 为元组变量, x 为 s 所基于的关系模式中的一个属性, c 是属性 x 所属域中的常量。

元组关系演算

根据如下规则用原子构造公式：

- 原子是公式；
- 如果 P_1 是公式，则 $\neg P_1$ 和 (P_1) 也都是公式；
- 如果 P_1 和 P_2 是公式，则 $P_1 \vee P_2$ 、 $P_1 \wedge P_2$ 和 $P_1 \rightarrow P_2$ 也都是公式；
- 如果 $P_1(s)$ 是包含自由元组变量 s 的公式， r 为关系，则 $\exists s \in r(P_1(s))$ 和 $\forall s \in r(P_1(s))$ 也都是公式。

元组关系演算中的等价性规则：

$$P_1 \wedge P_2 \iff \neg(\neg(P_1) \vee \neg(P_2)) \quad (28)$$

$$t \in r(P_1(t)) \iff \neg \exists t \in r(\neg P_1(t)) \quad (29)$$

$$P_1 \rightarrow P_2 \iff \neg(P_1) \vee P_2 \quad (30)$$

元组关系演算-查询示例

Example (例 2-30)

找出工资大于 80000 美元的所有教师的 ID

所有满足如下条件的元组 t 的集合：在关系 instructor 中存在元组 s 使 t 和 s 在属性 ID 上的值相等，且 s 在属性 salary 上的值大于 80000 美元。因为 ID 属性是对 t 进行限制的条件所涉及的唯一属性，因此结果只得到 ID 列上的关系。

$$\{t \mid \exists s \in \text{instructor}(t[\text{ID}] = s[\text{ID}] \wedge s[\text{salary}] > 80000)\}$$

Example (例 2-31)

找出位置在 Watson 楼的系中的所有教师姓名

元组变量 u 保证该系位于 Watson 楼，元组变量 s 被限制到与 u 的 deptname 相同。结果得到 name 列上的关系。

$$\{t \mid \exists s \in \text{instructor}(t[\text{name}] = s[\text{name}] \wedge \exists u \in \text{department}(u[\text{deptname}] = s[\text{deptname}] \wedge u[\text{building}] = \text{'Watson'}))\}$$

元组关系演算-查询示例

Example (例 2-32)

找出在 2024 年秋季学期或 2025 年春季学期或这两个学期都开设的所有课程的 courseid

给出至少满足下面两个条件之一的 courseid 元组的集合：

- 在关系 section 中满足 semester=Fall 且 year=2024 的某个元组包含该 courseid；
- 在关系 section 中满足 semester=Spring 且 year=2025 的某个元组包含该 courseid。

$$\{t | \exists s \in \text{section}(t[\text{courseid}] = s[\text{courseid}]) \wedge s[\text{semester}] = \text{"Fall"} \wedge s[\text{year}] = 2024) \\ \vee \exists u \in \text{section}(u[\text{courseid}] = t[\text{courseid}]) \wedge u[\text{semester}] = \text{"Spring"} \wedge u[\text{year}] = 2025)\}$$

Example (例 2-33)

找出只在 2024 年秋季和 2025 年春季两个学期都开设的所有课程的 courseid

$$\{t | \exists s \in \text{section}(t[\text{courseid}] = s[\text{courseid}]) \wedge s[\text{semester}] = \text{"Fall"} \wedge s[\text{year}] = 2024) \\ \wedge \exists u \in \text{section}(u[\text{courseid}] = t[\text{courseid}]) \wedge u[\text{semester}] = \text{"Spring"} \wedge u[\text{year}] = 2025)\}$$

元组关系演算-查询示例

Example (例 2-34)

找出 2024 年秋季开设而 2025 年春季不开的所有课程的 courseid

从 2024 年秋季开设的课程中去掉那些在 2025 年春季开设的课程

$$\{t | \exists s \in section(t[courseid] = s[courseid]) \wedge s[semester] = "Fall" \wedge s[year] = 2024) \wedge \\ \neg \exists u \in section(u[courseid] = t[courseid]) \wedge u[semester] = "Spring" \wedge u[year] = 2025\}$$

Example (例 2-35)

找出所有那些选了生物系全部课程的学生

所有满足如下条件的 ID 列上的元组 t 的集合：对关系 course 中所有元组 u，如果 u 在 deptname 属性上的值是 'Biology'，那么在关系 takes 中一定存在一个包含该学生 ID 以及该课程 courseid 的元组。

$$\{t | \exists r \in student(r[ID] = t[ID]) \wedge \forall u \in course(u[deptname] = "Biology" \rightarrow \\ \exists s \in takes(t[ID] = s[ID] \wedge s[courseid] = u[courseid])))\}$$

- 域 (domain)：元组关系公式 P 的域用 $\text{dom}(P)$ 表示，是 P 所引用的所有值的集合。它既包括 P 自身用到的值，又包括 P 中所涉及的关系的元组中出现的所有值。（i.e. P 中显式出现的值，以及名称出现在 P 中的那些关系的所有值的集合）。
- 如果出现在表达式 $t \mid P(t)$ 结果中的所有值均来自 $\text{dom}(P)$ ，则表达式是安全的，安全的表达式一定包含有限的结果。
- 域关系演算 (domain relational calculus)：关系演算的另一种形式，使用从属性域中取值的域变量，而不是整个元组的值。
- 形式化定义

$$\{ < x_1, x_2, \dots, x_n > \mid P(x_1, x_2, \dots, x_n) \} \quad (31)$$

其中

- x_1, x_2, \dots, x_n 分别是域变量, P 是由原子公式构成的公式
- 查询结果是所有包含 $< x_1, \dots, x_n >$ 的元组，并且 $< x_1, \dots, x_n >$ 使得公式 $P(x_1, \dots, x_n)$ 为真

原子可以为如下形式之一：

- $\langle x_1, x_2, \dots, x_n \rangle \in r$, 其中 r 为 n 个属性上的关系, x_1, x_2, \dots, x_n 为域变量或域常量;
- $x\theta y$, 其中 x 和 y 为域变量, θ 为比较运算符 (要求属性 x 和 y 所属域可用 θ 比较);
- $x\theta c$, 其中 x 为域变量, c 是 x 作为域变量的那个属性域中的常量。

根据如下规则用原子构造公式：

- 原子是公式;
- 如果 P_1 是公式, 则 $\neg P_1$ 和 (P_1) 也都是公式;
- 如果 P_1 和 P_2 是公式, 则 $P_1 \vee P_2$ 、 $P_1 \wedge P_2$ 和 $P_1 \rightarrow P_2$ 也都是公式;
- 如果 $P_1(x)$ 是包含自由域变量 x 的公式, 则 $\exists x(P_1(x))$ 和 $\forall x(P_1(x))$ 也都是公式。

把 $\exists a, b, c(P(a, b, c))$ 作为 $a(\exists b(\exists c(P(a, b, c))))$ 的简写。

域演算-查询示例

Example

找出工资在 80000 美元以上的教师的 ID、name、deptname 和 salary。

$$\{< i, n, d, s > \mid < i, n, d, s > \in instructor \wedge s > 80000\}$$

Example

找出工资大于 80000 美元的所有教师的姓名

$$\{< i > \mid \exists n, d, s (< i, n, d, s > \in instructor \wedge s > 80000)\}$$

域演算-查询示例

Example

找出在物理系的所有教师的姓名，以及他们教授的所有课程的 courseid。

$$\{< n, c > | \exists i, a, s, y (< I, c, a, s, y > \in teaches \wedge \exists d, s (< i, n, d, s > \in instructor \wedge d = "Physics"))\}$$

Example

找出在 2024 年秋季学期或 2025 年春季学期或这两个学期都开设的所有课程的集合

$$\{< c > | \exists a, s, y, b, r, t (< c, a, s, y, b, r, t > \in section \wedge s = "Fall" \wedge y = "2024") \vee \\ \exists a, s, y, b, r, t (< c, a, s, y, b, r, t > \in section \wedge s = "Spring" \wedge y = "2025")\}$$

1. 回顾

2. 关系模型

- 2.1. 关系数据结构
- 2.3. 关系完整性约束
- 2.4. 关系操作

3. 关系运算

3.1. 关系代数

- 传统的集合运算
- 专门的关系运算
- 关系代数查询综合

3.2. 关系演算

- 元组关系演算
- 元组关系演算

4. 参考文献

参考文献 |



B. E. F. Codd, "A relational model of data for large shared data banks," *M.D. computing : computers in medical practice*, vol. 15-3, pp. 162–166, 1970.

Thank you for your attention!
Q&A