第3章 关系运算

本章主要介绍关系数据模型的基本概念，关系运算和关系表达式的优化问题，其中关系运算和关系表达式的优化问题是本课程的重点内容之一。关系运算是关系数据模型的理论基础。

3.1 关系数据模型

3.1.1 关系数据模型的定义

用二维表格表示实体集，用关键码表示实体间联系的数据模型称为关系模型。

下面用集合代数来定义作为二维表格的关系。

定义 3.1 域(Domain)是值的集合。例如，学生性别的域是{男，女}，学生成绩的域是0～100的整数集合。

定义 3.2 给定一组域 D_1，D_2，…，D_n。D_1，D_2，…，D_n 上的笛卡儿积定义为集合：

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

其中每一个元素$(d_1, d_2, \cdots, d_n)$称为一个元组，元素中每一个值 d_i 称为元组分量。

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

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

例如，给出两个域，教师名域 D_1={汪宏伟，钱红}和课程名域 D_2={数据结构，离散数学，计算机原理}，那么 D_1 和 D_2 的笛卡儿积定义为集合：

$$D_1 \times D_2 = \{(\text{汪宏伟，数据结构}), (\text{汪宏伟，离散数学}), (\text{汪宏伟，计算机原理}),$$
$$(\text{钱红，数据结构}), (\text{钱红，离散数学}), (\text{钱红，计算机原理})\}$$

它表示教师名和课程名的所有可能的组合。其中，(汪宏伟，数据结构)，(汪宏伟，离散数学)，(汪宏伟，计算机原理)都是元组。汪宏伟、数据结构、离散数学都是分量。该笛卡儿积的基数为 2×3=6，也就是说，$D_1 \times D_2$ 一共有 2×3=6 个元组。

定义 3.3 域 D_1，D_2，…，D_n 上的笛卡儿积的子集称为在域 D_1，D_2，…，D_n 上的关系，用 $R(D_1, D_2, \cdots, D_n)$表示，这里 R 表示关系的名字，n 为关系的目或度(Arity)。关系的成员为元组，即笛卡儿积的子集的元素$(d_1, d_2, \cdots, d_n)$，值 d_i 为元组的第 i 个分量。例如，用教师名代替教师(假设无同名教师存在)，用课程名代替课程，教师任教的课程可用关系 TC(教师，课程)表示，它是教师名域和课程名域的笛卡儿积的子集，任一学期教师任课的记录

是这个关系的元组,比如:

TC={(汪宏伟,数据结构),(钱红,离散数学)}

TC 表示本学期汪宏伟老师上数据结构课程,钱红老师上离散数学课程。

还可以用集合论的观点来定义关系:关系是一个元数为 $k(k\geqslant 1)$ 的元组集合,即这个关系中有若干元组,每个元组有 k 个属性值。把关系看成是一个集合,集合中的元素是元组,即可将关系看成是一张二维表格。

表 3.1 是一张职工表,它是一张二维表格。

表 3.1 职工表(实体集)

职工编号	姓　名	部　门	性　别	年　龄	身份证号码
2113	程晓清	销售部	男	30	310110 **** 03062405
2116	刘红英	财务部	女	30	310110 **** 05082506
2136	李小刚	管理部	男	28	310110 **** 06092507
2138	蒋　民	采购部	男	41	310110 **** 08082405
2141	王国洋	销售部	男	39	310110 **** 09092407

从表 3.1 所示职工表的实例,可以归纳出关系具有如下特点。

(1) 关系(表)可以看成是由行和列(5 行和 6 列)交叉组成的二维表格。它表示的是一个实体集合。

(2) 表中一行称为一个元组,可用来表示实体集中的一个实体。

(3) 表中的列称为属性,给每一列起一个名称即属性名,表中的属性名不能相同。

(4) 列的取值范围称为域,同列具有相同的域,不同的列可有相同的域。例如,性别的取值范围是{男,女};职工编号和年龄都为整数域。

(5) 表中任意两行(元组)不能相同。能唯一标识表中不同行的属性或属性组称为主键。

尽管关系与二维表格、传统的数据文件有类似之处,但它们又有区别,严格地说,关系是一种规范化了的二维表格,具有如下性质。

(1) 属性值是原子的,不可分解。

(2) 没有重复元组。

(3) 没有行序。

(4) 理论上没有列序,为方便,使用时有列序。

3.1.2 关键码和表之间的联系

在关系数据库中,关键码(简称为键)是关系模型的一个重要概念。通常键由一个或几个属性组成,有如下四种键。

(1) 超键:在一个关系中,能唯一标识元组的属性或属性集称为关系的超键。

(2) 候选键:如果一个属性集能唯一标识元组,且又不含有多余的属性,那么这个属性集称为关系的候选键。

(3) 主键:若一个关系中有多个候选键,则选其中一个为关系的主键。用主键实现关系定义中“表中任意两行(元组)不能相同”的约束。包含在任何一个候选键中的属性称为主

属性(Primary Attribute),不包含在任何键中的属性称为非主属性(Nonprimary Attribute)或非键属性(Non-key Attribute)。

例如,表 3.1 的关系中,设属性集 k=(职工编号, 部门),虽然 k 能唯一地标识职工,但 k 只能是关系的超键,还不能作候选键使用。因为 k 中"部门"是一个多余属性,只有"职工编号"能唯一标识职工。因而"职工编号"是一个候选键。还有"身份证号"也可以是一个候选键。另外,如果规定"不允许有同名同姓的职工",那么"姓名"也可能是一个候选键。关系的候选键可以有多个,但不能同时使用,只能使用一个,例如使用"职工编号"来标识职工,那么"职工编号"就是主键了。

(4) 外键:若一个关系 R 中包含有另一个关系 S 的主键所对应的属性组 F,则称 F 为 R 的外键。并称关系 S 为参照关系,关系 R 为依赖关系。

例如,职工关系和部门关系分别为:

职工(职工编号, 姓名, 部门编号, 性别, 年龄, 身份证号码)

部门(部门编号, 部门名称, 部门经理)

职工关系的主键为职工编号,部门关系的主键为部门编号,在职工关系中,部门编号是它的外键。更确切地说,部门编号是部门表的主键,将它作为外键放在职工表中,实现两个表之间的联系。在关系数据库中,表与表之间的联系就是通过公共属性实现的。一般在主键的属性下面加下画线,在外键的属性下面加波浪线。

3.1.3 关系模式、关系子模式和存储模式

关系模型基本上遵循数据库的三级体系结构。在关系模型中,概念模式是关系模式的集合,外模式是关系子模式的集合,内模式是存储模式的集合。

1. 关系模式

关系模式是对关系的描述,它包括模式名、组成该关系的诸属性名、值域名和模式的主键。具体的关系称为实例。

【例 3.1】 图 3.1 是一个教学模型的实体联系图。实体类型"学生"的属性 SNO、SNAME、AGE、SEX、SDEPT 分别表示学生的学号、姓名、年龄、性别和学生所在系;实体类型"课程"的属性 CNO、CNAME、CDEPT、TNAME 分别表示课程号、课程名、课程所属系和任课教师。学生用 S 表示,课程用 C 表示。S 和 C 之间有 M∶N 的联系(一个学生可选多门课程,一门课程可以被多个学生选修),联系类型 SC 的属性成绩用 GRADE 表示。图 3.1 表示的实体联系图(ER 图)转换成关系模式集如图 3.2 所示。ER 图向关系模型的转换技术将在第 7 章中做详细介绍。表 3.2 是这个关系模式的实例。

又如图 3.1 表示的是学生关系的基本情况,相应的关系模式为:

```
S(SNO, SNAME, AGE, SEX, SDEPT)
```

这个关系模式描述了学生的数据结构,它是图 3.1 中学生关系(表格)的关系模式。

关系模式是用数据定义语言(DDL)定义的。关系模式的定义包括:模式名、属性名、值域名以及模式的主键。由于不涉及物理存储方面的描述,因此关系模式仅仅是对数据本身的特征的描述。

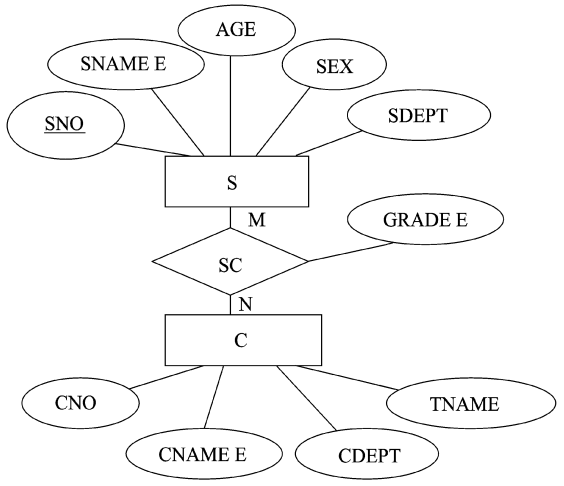

图 3.1　实体联系图

学生关系模式S(SNO, SNAME, AGE, SEX, SDEPT)
学习关系模式SC(SNO, CNO, GRADE)
课程关系模式C(CNO, CNAME, CDEPT, TNAME)

图 3.2　关系模式集

表 3.2　三个关系

SNO	SNAME	AGE	SEX	SDEPT
S1	程　宏	19	男	计算机
S3	刘莎莎	18	女	通信
S4	李刚畸	20	男	法学
S6	蒋天云	19	男	国际贸易
S9	王　莉	21	女	计算机

(a) 学生关系

CNO	CNAME	CDEPT	TNAME
C2	离散数学	计算机	汪宏伟
C3	高等数学	通信	钱　红
C4	数据结构	计算机	马　良
C1	计算机原理	计算机	李　兵

(b) 课程关系

SNO	CNO	GRADE
S3	C3	87
S1	C2	88
S4	C3	79
S9	C4	83
S1	C3	76
S6	C3	68
S1	C1	78
S6	C1	88
S3	C2	64
S1	C4	86
S9	C2	78

(c) 学习关系

2. 关系子模式

有时,用户使用的数据不是直接来自关系模式中的数据,而是从若干关系模式中抽取满足一定条件的数据。这种结构可用关系子模式实现。关系子模式是用户所需数据的结构的描述,其中包含这些数据来自哪些模式和应满足哪些条件。

【例 3.2】 用户需要用到成绩子模式 G(SNO, SNAME, CNO, GRADE)。子模式 G 对应的数据来源于表 S 和表 SC,构造时应满足它们的 SNO 值相等。子模式 G 的构造过程如图 3.3 所示。

子模式定义语言还可以定义用户对数据进行操作的权限,例如是否允许读、修改等。由于关系子模式来源于多个关系模式,因此是否允许对子模式的数据进行插入和修改就不一定了。

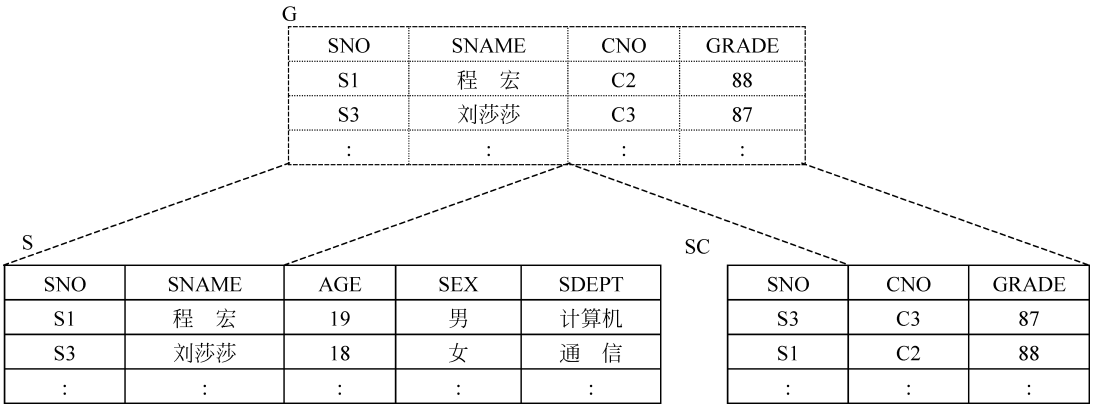

图 3.3 子模式 G 的定义

3. 存储模式

存储模式描述了关系是如何在物理存储设备上存储的。关系存储时的基本组织方式是文件。由于关系模式有关键码，因此存储一个关系可以用散列方法或索引方法实现。如果关系中元组数目较少(100 以内)，那么也可以用堆文件方式实现。此外，还可以对任意的属性集建立辅助索引。

3.1.4 关系模型的完整性规则

关系模型的完整性规则是对数据的约束。关系模型提供了三类完整性规则：实体完整性规则、参照完整性规则、用户定义的完整性规则。其中实体完整性规则和参照完整性规则是关系模型必须满足的完整性的约束条件，称为关系完整性规则。

1. 实体完整性规则

完整性约束条件示例如图 3.4 所示。在图 3.4 中给出导师表和研究生表，其中导师表的主键是导师编号，研究生表的主键是学号，这两个主键的值在表中是唯一的和确定的，这样才能有效地标识每一个导师和研究生。主键不能取空值(NULL)，空值不是 0，也不是空字符串，是没有值，或是不确定的值，所以空值无法标识表中的一行。为了保证每一个实体有唯一的标识符，主键不能取空值。

实体完整性规则：关系中元组的主键值不能为空。

例如，图 3.4 所示研究生表的主键是学号，不包含空的数据项；导师表的主键是导师编号，也不包含空的数据项，所以，这两个表都满足实体完整性规则。

2. 参照完整性规则

在关系数据库中，关系与关系之间的联系是通过公共属性实现的。这个公共属性是一个表的主键和另一个表的外键。外键必须是另一个表的主键的有效值，或者是一个“空值”。例如，图 3.4 中研究生表与导师表之间的联系是通过导师编号实现的，导师编号是导师表的主键、研究生表的外键。研究生表中的导师编号必须是导师表中导师编号的有效值，或者“空值”，否则，就是非法的数据。从图 3.4 所示的研究生表中，可以看到学号为“S107”的研究生没有固定的导师，所以他的导师编号为“空值”；而学号为“S110”的研究生的导师编号为“318”，由于导师表中不存在导师编号“318”，所以这个值是非法的。

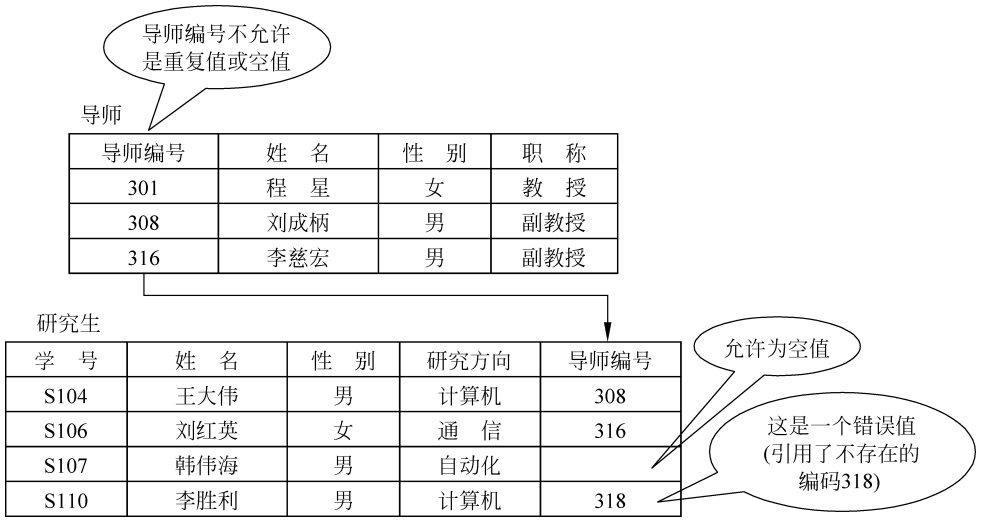

图 3.4 完整性约束条件示例

参照完整性规则的形式定义如下：

如果属性集 K 是关系模式 R1 的主键，同时也是关系模式 R2 的外键，那么在 R2 的关系中，K 的取值只允许两种可能，或者为空值，或者等于 R1 关系中某个主键值。

这条规则在使用时，有以下三点须注意。

(1) 外键和相应的主键可以不同名，只要定义在相同值域上即可。

(2) R1 和 R2 也可以是同一个关系模式，表示了同一个关系中不同元组之间的联系。例如表示课程之间先修联系的模式 R(CNO, CNAME, PCNO)，其属性表示课程号、课程名、先修课程的课程号，R 的主键是 CNO，而 PCNO 就是一个外键，表示 PCNO 值一定要在关系中存在(某个 CNO 值)。

(3) 外键值是否允许空，应视具体问题而定。在模式中，若外键是该模式主键中的成分时，则外键值不允许空，否则允许空。

在上述形式定义中，R1 称为“参照关系”模式，R2 称为“依赖关系”模式。在软件开发工具 PowerBuilder 中，分别称为主表和副表；在 Visual FoxPro 系统中，分别称为父表和子表。

上述两类完整性规则是关系模型必须满足的规则，应该由系统自动支持。

3. 用户定义的完整性规则

这是针对某一具体数据的约束条件，由应用环境决定。它反映某一具体应用所涉及的数据必须满足的语义要求。系统应提供定义和检验这类完整性的机制，以便用统一的系统方法处理它们，不再由应用程序承担这项工作。例如学生成绩应该大于或等于 0，职工的工龄应小于年龄，人的身高不能超过 3m 等。

3.1.5 关系模型的形式定义

关系模型有三个组成部分：数据结构、数据操作、完整性规则。

(1) 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。关系模型

基本的数据结构是关系。

(2) 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。关系运算分为关系代数和关系演算两类。

(3) 关系模型的三类完整性规则。

3.2 关系代数

3.2.1 关系查询语言和关系运算

关系数据库的数据操纵语言(DML)的语句分为查询语句和更新语句两大类。查询语句用于描述用户的各类检索要求;更新语句用于描述用户的插入、修改和删除等操作。

关系查询语言根据其理论基础的不同分为两大类。

(1) 关系代数语言:查询操作是以集合操作为基础的运算。

(2) 关系演算语言:查询操作是以谓词演算为基础的运算。

关系查询语言是一种比 Pascal、C 等程序设计语言更高级的语言。Pascal、C 一类语言属于过程性(Procedural)语言,在编程时必须给出获得结果的操作步骤,即指出"干什么"和"怎么干"。关系查询语言属于非过程(Nonprocedural)语言,编程时只需指出需要什么信息,而不必给出具体的操作步骤,即只要指出"干什么",而不必指出"怎么干"。

各类关系查询语言均属于非过程性语言,但其"非过程性"的强弱程度不一样。关系代数语言的非过程性较弱,在查询表达式中必须指出操作的先后顺序;关系演算语言的非过程性较强,操作顺序仅限于量词的顺序。

3.2.2 关系代数的五个基本操作

关系代数是以集合代数为基础发展起来的,它是以关系为运算对象的一组高级运算的集合。

由于关系定义为元数相同的元组集合,因此把关系看成集合,集合代数中的操作(并、差、交、笛卡儿积)就可以引入到关系运算中来。还有一些操作是针对关系数据库环境专门设计的,比如对关系进行垂直分割(投影)、水平分割(选择)、关系的结合(连接)等。

关系代数有五个基本的操作。

1. 并(Union)

设关系 R 和关系 S 具有相同的元数 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 和关系 S 的并由属于 R 或属于 S 的元组组成,其结果仍为 n 元的关系,记为 R∪S。形式定义如下:

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

其中,t 是元组变量,R 和 S 的元数相同。两个关系的并运算是将两个关系中的所有元组构成一个新关系。并运算要求两个关系属性的性质必须一致且并运算的结果要消除重复的元组。

【例 3.3】 有库存和进货两个关系如表 3.3 所示,要将两个关系合并为一个关系,可用并运算实现。

表 3.3　关系代数的并运算

商品编号	品名	数量
2008230	冰箱	19
2008234	彩电	50
2007156	空调	20

(a) 库存关系

商品编号	品名	数量
2008124	电熨斗	30
2008310	微波炉	18

(b) 进货关系

商品编号	品名	数量
2008230	冰箱	19
2008234	彩电	50
2007156	空调	20
2008124	电熨斗	30
2008310	微波炉	18

(c) 并运算结果

2. 差(Difference)

设关系 R 和关系 S 具有相同的元数 n,且相应的属性取自同一个域,则关系 R 和 S 的差由属于 R 而不属于 S 的所有元组组成。其结果仍为 n 元的关系。记为 R−S。形式定义如下:

$$R-S \equiv \{t \mid t \in R \wedge t \overline{\in} S\}$$

其中,t 是元组变量,R 和 S 的元数相同。

【例 3.4】 有考生成绩合格者名单和身体不合格者名单两个关系,按录取条件将成绩合格且身体健康的考生产生录取名单关系。这个任务可以用差运算来完成如表 3.4 所示。

表 3.4　关系代数的差运算

考生号
20013211
20011231
20017156
20018124
20013610

(a) 成绩合格考生

考生号
20013211
20017156
20013610

(b) 身体不合格考生

考生号
20011231
20018124

(c) 差运算结果

3. 笛卡儿积(Cartesian Product)

设关系 R 和关系 S 的元数分别为 r 和 s。定义 R 和 S 的笛卡儿积 R×S 是一个($r+s$)元的元组集合,每个元组的前 r 个分量(属性值)来自 R 的一个元组,后 s 个分量是 S 的一个元组,记为 R×S。形式定义如下:

$$R \times S \equiv \{t \mid t = \langle t^r, t^s \rangle \wedge t^r \in R \wedge t^s \in S\}$$

其中,t^r、t^s 中 r、s 为上标,分别表示有 r 个分量和 s 个分量,若 R 有 n 个元组,S 有 m 个元组,则 R×S 有 $n \times m$ 个元组。

【例 3.5】 在学生和必修课程两个关系上,产生选修关系:要求每个学生必须选修所有必修课程。这个选修关系可以用两个关系的笛卡儿积运算来实现如表 3.5 所示。

4. 投影(Projection)

这个操作是对一个关系进行垂直分割,消去某些列,并重新安排列的顺序,再删去重复元组。

设关系 R 是 k 元关系,R 在其分量 $A_{i_1}, A_{i_2}, \cdots, A_{i_m}$($m \leqslant k, i_1, i_2, \cdots, i_m$ 为 1～k 的整数)上的投影用 $\pi_{i_1, i_2, \cdots, i_m}(R)$ 表示,它是从 R 中选择若干属性列组成的一个 m 元元组的集合,形式定义如下:

$$\pi_{i_1, i_2, \cdots, i_m}(R) \equiv \{t \mid t = \langle t_{i_1}, t_{i_2}, \cdots, t_{i_m} \rangle \wedge \langle t_1, t_2, \cdots, t_k \rangle \in R\}$$

表 3.5　关系代数的笛卡儿积运算

SNO	SNAME
S1	程　宏
S3	刘莎莎
S4	李刚畸

(a) 学生关系

CNO	CNAME	CREDIT
C4	数据结构	6
C1	计算机原理	6
C3	高等数学	8

(b) 课程关系

SNO	SNAME	CNO	CNAME	CREDIT
S1	程　宏	C4	数据结构	6
S1	程　宏	C1	计算机原理	6
S1	程　宏	C3	高等数学	8
S3	刘莎莎	C4	数据结构	6
S3	刘莎莎	C1	计算机原理	6
S3	刘莎莎	C3	高等数学	8
S4	李刚畸	C4	数据结构	6
S4	李刚畸	C1	计算机原理	6
S4	李刚畸	C3	高等数学	8

(c) 学习关系

【例 3.6】 已知职工表如表 3.1 所示，对职工表进行投影操作。

(1) 列出所有职工的职工编号、姓名、部门。关系代数表示为：

$$\pi_{职工编号,姓名,部门}(职工)$$

结果如表 3.6 所示。

(2) 列出职工表中的所有部门，关系代数表示为：

$$\pi_{部门}(职工)$$

结果如表 3.7 所示。

注意，由于投影的结果消除了重复元组，所以，结果只有 4 个元组。

表 3.6　关系代数的投影运算实例一

职工编号	姓　名	部　门
2113	程晓清	销售部
2116	刘红英	财务部
2136	李小刚	管理部
2138	蒋　民	采购部
2141	王国洋	销售部

表 3.7　关系代数的投影运算实例二

部门
销售部
财务部
管理部
采购部

5. 选择(Selection)

这个操作是根据某些条件对关系进行水平分割，即选择符合条件的元组。条件用命题公式 F 表示，F 中的运算对象是常量(用引号括起来)或元组分量(属性名或列的序号)，运算符有算术比较运算符(<、≤、>、≥、=、≠，这些符号统称为 θ 符)和逻辑运算符(∧、∨、¬)。

关系 R 关于公式 F 的选择操作用 $\sigma_F(R)$ 表示，形式定义如下：

$$\sigma_F(R)=\{t \mid t \in R \wedge F(t)=true\}$$

其中，σ 为选择运算符，$\sigma_F(R)$ 表示从 R 中挑选满足公式 F 的元组所构成的关系。

【例 3.7】 已知学生表 S 如表 3.2 所示，对学生表进行选择操作：列出所有女同学的基本情况。选择的条件是：SEX='女'。用关系代数表示为：$\sigma_{SEX='女'}(S)$，也可以用属性序号表示属性名：$\sigma_{4='女'}(S)$，结果如表 3.8 所示。

表 3.8　关系代数的选择运算

SNO	SNAME	AGE	SEX	SDEPT
S3	刘莎莎	18	女	通信
S9	王　莉	21	女	计算机

3.2.3　关系代数的组合操作

3.2.2 节所述五个基本操作可以组合成下列四个操作。

1. 交(Intersection)

设关系 R 和关系 S 具有相同的元数 n(即两个关系都有 n 个属性),而且相应的属性取自同一个域。关系 R 和 S 的交记为 R∩S,结果仍为 n 元的关系。由既属于 R 又属于 S 的元组组成。形式定义如下:

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

其中,t 是元组变量,R 和 S 的元数相同。关系的交可以由关系的差来表示,即:

$$R \cap S \equiv R-(R-S) \quad 或 \quad R \cap S \equiv S-(S-R)$$

【例 3.8】 假设有优秀学生和优秀学生干部两个表如表 3.9(a)、(b)所示。要求检索既是优秀学生又是优秀学生干部的学生。这个检索可以用交操作来实现。结果如表 3.9(c)所示。

表 3.9　关系代数的交运算

SNO	SNAME
S3	刘莎莎
S4	李刚畸
S5	李小刚
S8	姜　名
S19	王　燕

(a) 优秀学生关系 S1

SNO	SNAME
S1	程　宏
S4	李刚畸
S5	李小刚
S7	柳庆国

(b) 优秀学生干部关系 S2

SNO	SNAME
S4	李刚畸
S5	李小刚

(c) 新关系(S1∩S2)

2. 连接(Join)

连接操作可将两个关系连在一起,形成一个新的关系。连接操作是笛卡儿积和选择操作的组合。连接分为 θ 连接和 F 连接两种。

(1) θ 连接:θ 连接是从关系 R 和 S 的笛卡儿积(R×S)中选取属性值满足某一 θ 操作的元组,记为 $R \underset{i\theta j}{\bowtie} S$,这里 i 和 j 分别是关系 R 和 S 中的第 i 个、第 j 个属性的序号,θ 是算术比较符。形式定义如下:

$$R \underset{i\theta j}{\bowtie} S \equiv \sigma_{i\theta(r+j)}(R \times S)$$

其中,r 是关系 R 的元数。该式表示 θ 连接是在关系 R 和 S 的笛卡儿积中挑选第 i 个分量和第$(r+j)$个分量满足 θ 运算的元组。如果 θ 为等号"=",那么这个连接操作称为等值连接。

(2) F 连接:F 连接是从关系 R 和 S 的笛卡儿积中选取属性间满足某一公式 F 的元组,记为 $R \underset{F}{\bowtie} S$。这里 F 是形为 $F_1 \wedge F_2 \wedge \cdots \wedge F_n$ 的公式,每个 F_i 是形为 $i\theta j$ 的式子。而 i

和 j 分别为关系 R 和 S 的第 i 个分量和第 j 个分量的序号。

【例 3.9】 表 3.10(a)、(b)、(c)分别是关系 SC、C 和 CL，表 3.10(d)是 $SC \underset{2=1}{\bowtie} C$ 的值，表 3.10(e)是 $SC \underset{2=1 \wedge 3>2}{\bowtie} CL$ 的值。

表 3.10　关系代数的连接运算

SNO	CNO	GRADE
S3	C3	87
S1	C2	88
S4	C3	79
S1	C3	76
S5	C2	91
S6	C1	78

(a) 关系 SC

CNO	CNAME	CDEPT	TNAME
C2	离散数学	计算机	汪宏伟
C3	高等数学	通信	钱　红
C4	数据结构	计算机	马　良
C1	计算机原理	计算机	李　兵

(b) 关系 C

CNO	G	LEVEL
C2	85	A
C3	85	A

(c) 关系 CL

SNO	SC. CNO	GRADE	C. CNO	CNAME	CDEPT	TNAME
S3	C3	87	C3	高等数学	通信	钱　红
S1	C2	88	C2	离散数学	计算机	汪宏伟
S4	C3	79	C3	高等数学	通信	钱　红
S1	C3	76	C3	高等数学	通信	钱　红
S5	C2	91	C2	离散数学	计算机	汪宏伟
S6	C1	78	C1	计算机原理	计算机	李　兵

(d) $SC \underset{2=1}{\bowtie} C$

SNO	SC. CNO	GRADE	CL. CNO	G	LEVEL
S3	C3	87	C3	85	A
S1	C2	88	C2	85	A
S5	C2	91	C2	85	A

(e) $SC \underset{2=1 \wedge 3>2}{\bowtie} CL$

3. 自然连接(Natural Join)

自然连接是一种特殊的等值连接，它要求两个关系中进行比较的分量必须是相同的属性组，并且要在结果中把重复的属性去掉。两个关系的自然连接用 $R \bowtie S$ 表示，具体计算过程如下：

(1) 计算 $R \times S$。

(2) 设 R 和 S 的公共属性是 $A_1, A_2, \cdots, A_K$，挑选 $R \times S$ 中满足 $R.A_1 = S.A_1, \cdots, R.A_K = S.A_K$ 的那些元组。

(3) 去掉 $S.A_1, S.A_2, \cdots, S.A_K$ 这些列(保留 $R.A_1, R.A_2, \cdots, R.A_K$)。

因此，$R \bowtie S$ 可用下式定义：

$$R \bowtie S \equiv \pi_{i_1, i_2, \cdots, i_m}(\sigma_{R.A_1 = S.A_1 \wedge \cdots \wedge R.A_K = S.A_K}(R \times S))$$

【例 3.10】 表 3.11(c)表示关系 SC 和 C 的自然连接，这里

$$SC \bowtie C \equiv \pi_{SNO, SC.CNO, GRADE, CNAME, CDEPT, TNAME}(\sigma_{SC.CNO = C.CNO}(SC \times C))$$

表 3.11　关系代数的自然连接运算

SNO	CNO	GRADE
S3	C3	87
S1	C2	88
S4	C3	79
S1	C3	76
S5	C2	91
S6	C1	78

(a) 关系 SC

CNO	CNAME	CDEPT	TNAME
C2	离散数学	计算机	汪宏伟
C3	高等数学	通信	钱　红
C4	数据结构	计算机	马　良
C1	计算机原理	计算机	李　兵

(b) 关系 C

SNO	CNO	GRADE	CNAME	CDEPT	TNAME
S3	C3	87	高等数学	通信	钱　红
S1	C2	88	离散数学	计算机	汪宏伟
S4	C3	79	高等数学	通信	钱　红
S1	C3	76	高等数学	通信	钱　红
S5	C2	91	离散数学	计算机	汪宏伟
S6	C1	78	计算机原理	计算机	李　兵

(c) SC ⋈ C

自然连接是构造新关系的有效方法,是关系代数中常用的一种运算,在关系数据库理论中起着重要作用。利用投影、选择和自然连接操作可以任意地分解和构造关系。

4. 除(Division)

设两个关系 R 和 S 的元数分别为 r 和 s(设 $r>s>0$),那么 R÷S 是一个($r-s$)元的元组的集合。R÷S 是满足下列条件的最大关系:其中每个元组 t 与 S 中每个元组 u 组成的新元组〈t,u〉必在关系 R 中。为方便起见,这里假设 S 的属性为 R 中后 s 个属性。

R÷S 的具体计算过程如下:

(1) $T=\pi_{1,2,\cdots,r-s}(R)$

(2) $W=(T\times S)-R$

(3) $V=\pi_{1,2,\cdots,r-s}(W)$

(4) $R\div S=T-V$

即 $R\div S\equiv\pi_{1,2,\cdots,r-s}(R)-\pi_{1,2,\cdots,r-s}((\pi_{1,2,\cdots,r-s}(R)\times S)-R)$。

【例 3.11】 表 3.12(a)表示学生学习关系 SC,表 3.12(b)表示课程成绩条件关系 CG,表 3.12(c)表示满足课程成绩条件(离散数学为优和数据结构为优)的学生情况关系,用(SC÷CG)表示。

3.2.4　关系代数表达式及其应用实例

在关系代数运算中,由五种基本操作经过有限次复合的表达式称为关系代数表达式。这种表达式的运算结果仍是一个关系,可以用关系代数表达式表示各种数据查询操作。

表 3.12　关系代数的除法运算

SNAME	SEX	CNAME	CDEPT	GRADE
李志鸣	男	离散数学	通信	优
刘月莹	女	离散数学	计算机	良
吴　康	男	离散数学	通信	优
王文晴	女	数据结构	计算机	优
吴　康	男	高等数学	通信	良
王文晴	女	离散数学	计算机	优
刘月莹	女	数据结构	计算机	优
李志鸣	男	数据结构	通信	优
李志鸣	男	高等数学	通信	良

(a) 学生学习关系 SC

CNAME	GRADE
离散数学	优
数据结构	优

(b) 课程成绩条件关系 CG

SNAME	SEX	CDEPT
李志鸣	男	通信
王文晴	女	计算机

(c) SC÷CG

查询语句的关系代数表达式的一般形式是：

$$\pi\cdots(\sigma\cdots(R\times S))$$

或者

$$\pi\cdots(\sigma\cdots(R\bowtie S))$$

上面的式子表示：首先取得查询涉及的关系，再执行笛卡儿积或自然连接操作得到一张中间表格，然后对该中间表格执行水平分割(选择操作)和垂直分割(投影操作)，见例 3.12 的(2)～(5)。当查询涉及否定或全部、包含值时，上述形式就不能表达了，就要用到差操作或除法操作，见例 3.12 的(6)～(8)。

【例 3.12】 设供应商、零件、工程之间的供应关系是一个三元联系，其 ER 图的原型见第 2 章的图 2.9。这个 ER 图转换成关系模式有四个，其结构如下所示。

供应商关系：S(SNO, SNAME, SADDR)

零件关系：P(PNO, PNAME, COLOR, WEIGHT)

工程项目关系：J(JNO, JNAME, JCITY, BALANCE)

供应情况关系：SPJ(SNO,PNO,JNO, PRICE, QTY)

上述各属性的含义：供应商编号(SNO)、供应商名称(SNAME)、供应商地址(SADDR)、零件编号(PNO)、零件名称(PNAME)、颜色(COLOR)、重量(WEIGHT)、工程项目编号(JNO)、工程名称(JNAME)、工程所属城市(JCITY)、工程项目余额(BALANCE)、零件单价(PRICE)、供应数量(QTY)。

试用关系代数表达式表示每个查询语句。

(1) 检索供应零件给工程 J1 的供应商编号 SNO 与零件编号 PNO。

$$\pi_{SN0,\ PN0}(\sigma_{JNO='J1'}(SPJ))$$

该式表示先对关系 SPJ 执行选择操作，然后执行投影操作。另外，表达式中也可以不写属性名，而写属性的序号：

$$\pi_{1,2}(\sigma_{3='J1'}(SPJ))$$

(2) 检索供应零件给工程 J1，且零件编号为 P1 的供应商编号 SNO。

$$\pi_{SNO}(\sigma_{JNO='J1'\wedge PNO='P1'}(SPJ))$$

(3) 检索使用了编号为 P3 零件的工程编号和名称。

$$\pi_{JNO,JNAME}(\sigma_{PNO='P3'}(J \bowtie SPJ))$$

这个查询涉及关系 J 和 SPJ,因此先要对这两个关系执行自然连接操作,然后再对其执行选择和投影操作。

(4) 检索供应零件给工程 J1,且零件颜色为红色的供应商名称 SNAME 和地址 SADDR。

$$\pi_{SNAME,\ SADDR}(\sigma_{JNO='J1' \wedge COLOR='红色'}(S \bowtie SPJ \bowtie P))$$

(5) 检索使用了零件编号为 P3 或 P5 零件的工程编号 JNO。

$$\pi_{JNO}(\sigma_{PNO='P3' \vee PNO='P5'}(SPJ))$$

(6) 检索至少使用了编号为 P3 和 P5 零件的工程编号 JNO。

$$\pi_{3}(\sigma_{3=8 \wedge 2='P3' \wedge 7='P5'}(SPJ \times SPJ))$$

这里(SPJ×SPJ)表示关系 SPJ 自身相乘的笛卡儿积操作。这里的 σ 是对关系(SPJ×SPJ)进行选择操作,其中的条件(3=8∧2='P3'∧7='P5')表示同一个工程,既使用了零件 P3 又使用了零件 P5。

(7) 检索不使用编号为 P3 零件的工程编号 JNO 和工程名称 JNAME。

$$\pi_{JNO,JNAME}(J) - \pi_{JNO,JNAME}(\sigma_{PNO='P3'}(J \bowtie SPJ))$$

这里要用到集合差操作。先检索全部工程号,再检索使用了编号为 P3 零件的工程,最后执行两个集合的差操作。

(8) 检索使用了全部零件的工程名称 JNAME。

编写这个查询语句的关系代数表达式的过程如下:

① 工程使用零件情况可用操作 $\pi_{JNO,PNO}(SPJ)$表示。

② 全部零件用操作 $\pi_{PNO}(P)$表示。

③ 使用了全部零件的工程编号可用除法操作表示,操作结果是工程编号 JNO 集:

$$(\pi_{JNO,PNO}(SPJ) \div \pi_{PNO}(P))$$

④ 根据 JNO 检索工程名称 JNAME,可以用自然连接和投影操作组合而成:

$$\pi_{JNAME}(J \bowtie (\pi_{JNO,PNO}(SPJ) \div \pi_{PNO}(P)))$$

(9) 检索使用零件包含编号为 S1 的供应商所供应的全部零件的工程编号 JNO。

① 工程使用 S1 供应商所提供的零件情况可用操作 $\pi_{JNO,PNO}(\sigma_{SNO='S1'}(SPJ))$表示。

② 编号为 S1 的供应商所供应的全部零件情况可用操作 $\pi_{PNO}(\sigma_{SNO='S1'}(SPJ))$表示。

③ 使用零件包含编号为 S1 的供应商所供应的全部零件的工程编号,可以用除法操作表示:

$$\pi_{JNO,PNO}(\sigma_{SNO='S1'}(SPJ)) \div \pi_{PNO}(\sigma_{SNO='S1'}(SPJ))$$

3.2.5 扩充的关系代数操作

为了在关系代数操作时多保存一些信息,下面引进“外连接”和“外部并”两种操作。

1. 外连接(Outer Join)

在关系 R 和关系 S 做自然连接时,选择两个关系在公共属性上值相等的元组构成新关系的元组。此时,关系 R 中某些元组有可能在关系 S 中不存在公共属性上值相等的元组,造成关系 R 中这些元组的值在操作时被舍弃。由于同样的原因,关系 S 中某些元组也有可能被舍弃。为了在操作时能保存这些将被舍弃的元组,提出了“外连接”操作。

如果关系 R 和关系 S 做自然连接时，把原该舍弃的元组也保留在新关系中，同时在这些元组新增加的属性上填上空值(Null)，这种操作称为“外连接”操作，用符号 R ⟗ S 表示。

如果关系 R 和关系 S 做自然连接时，只把关系 R 中原该舍弃的元组放到新关系中，那么这种操作称为“左外连接”操作，用符号 R ⟕ S 表示。

如果关系 R 和关系 S 做自然连接时，只把关系 S 中原该舍弃的元组放到新关系中，那么这种操作称为“右外连接”操作，用符号 R ⟖ S 表示。

【例 3.13】 表 3.13 表示关系代数的外连接运算。

表 3.13 关系代数的外连接运算

A	B	C
a	b	c
b	b	f
c	a	d

(a) 关系 R

B	C	D
b	c	d
b	c	e
a	d	b
e	f	g

(b) 关系 S

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b

(c) R ⋈ S

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null
null	e	f	g

(d) R ⟕ S

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
b	b	f	null

(e) R ⟖ S

A	B	C	D
a	b	c	d
a	b	c	e
c	a	d	b
null	e	f	g

(f) R ⟗ S

2. 外部并(Outer Union)

前面定义两个关系的并操作时，要求关系 R 和关系 S 具有相同的关系模式。如果关系 R 和关系 S 的关系模式不同，构成的新关系属性由关系 R 和关系 S 的所有属性组成(公共属性只取一次)，新关系的元组由属于关系 R 或属于关系 S 的元组构成，同时元组在新增加的属性上填上空值，那么这种操作称为外部并操作。

【例 3.14】 表 3.14 是表 3.13 中关系 R 和关系 S 执行外部并后的结果。

表 3.14 关系代数的外部并运算

A	B	C	D	A	B	C	D
a	b	c	null	null	b	c	e
b	b	f	null	null	a	d	b
c	a	d	null	null	e	f	g
null	b	c	d				

在分布式数据库中还经常用到下面一种“半连接”操作。

3. 半连接(Semijoin)

关系 R 和关系 S 的半连接操作记为 R ⋉ S，定义为关系 R 和关系 S 的自然连接在关系

R 的属性集上的投影,即 $R \ltimes S \equiv \pi_R(R \bowtie S)$。

这里 π_R 的下标 R 表示关系 R 的属性集。

也可以用另一种方法计算 $R \ltimes S$:先求出关系 S 在关系 R 和关系 S 的公共属性集上的投影,再求关系 R 和这个投影的自然连接,即 $R \ltimes S = R \bowtie \pi_{R \cap S}(S)$。显然,半连接的交换律是不成立的,即 $R \ltimes S \neq S \ltimes R$。

【例 3.15】 表 3.15 是两个关系做自然连接和半连接的例子。

表 3.15 关系代数的半连接运算

A	B	C
a	b	c
d	b	c
b	b	f
c	a	d

(a) 关系 R

B	C	D
b	c	d
b	c	e
a	d	b

(b) 关系 S

A	B	C	D
a	b	c	d
a	b	c	e
d	b	c	d
d	b	c	e
c	a	d	b

(c) $R \bowtie S$

A	B	C
a	b	c
d	b	c
c	a	d

(d) $R \ltimes S$

B	C	D
b	c	d
b	c	e
a	d	b

(e) $S \ltimes R$

*3.3 关系演算

关系演算运算是以数理逻辑中的谓词演算为基础,用公式表示关系运算的条件。关系演算根据所用到的变量不同可分为元组关系演算和域关系演算,前者以元组为变量,后者以域为变量,分别简称为元组演算和域演算。

3.3.1 元组关系演算

1. 原子公式和公式的定义

元组关系演算用表达式{t|P(t)}表示。其中 t 是元组变量,表示一个定长的元组;P 是公式,公式由原子公式组合而成。

定义 3.4 原子公式(Atoms)有下列三种形式。

(1) R(s):R 是关系名,s 是元组变量。R(s)表示这样一个命题:s 是关系 R 的一个元组。所以,关系 R 可表示为{s|R(s)}。

(2) s[i]θu[j]:s 和 u 是元组变量,θ 是算术比较运算符。s[i]θu[j]表示这样一个命题:元组 s 的分量 i 与元组 u 的分量 j 满足比较关系 θ。例如 s[2]>u[1]表示元组 s 的第二个分量必须大于元组 u 的第一个分量。

(3) s[i]θc 或 cθu[j]:s 和 u 是元组变量,c 是常量。s[i]θc 或 cθu[j]表示元组 s(或 u)的第 i 个(或第 j 个)分量与常量 c 满足比较关系 θ。例如 s[3]='5'表示元组 s 的第三个分量值为 5。

在定义关系演算操作时,要用到“自由”(Free)或“约束”(Bound)变量概念。在一个公式中,如果元组变量的前面没有用到存在量词∃或全称量词∀等符号,那么称之为自由元组变量,否则称之为约束元组变量。约束变量类似于程序设计语言中过程内部定义的局部变量,自由变量类似于过程外部定义的外部变量或全局变量。

定义 3.5 公式(Formulas)、公式中的自由元组变量、约束元组变量按下列方式递归定义：

(1) 每个原子公式是一个公式。其中的元组变量是自由变量。

(2) 如果 P1 和 P2 是公式，那么 P1∧P2、P1∨P2、¬ P1 和 P1=>P2 也是公式，分别表示如下命题："P1 和 P2 同时为真"；"P1 和 P2 中的一个或同时为真"；"P1 为假"；"若 P1 为真，则 P2 为真"。公式中的变量是自由的还是约束的，同在 P1 和 P2 中一样。

(3) 如果 P 是公式，那么(∃t)(P)也是公式。(∃t)(P)表示这样一个命题："存在一个元组 t 使得公式 P 为真"。元组变量 t 在 P 中是自由的，在(∃t)(P)中是约束的。P 中其他元组是自由的或约束的，在(∃t)(P)中没有变化。

(4) 如果 P 是公式，则(∀t)(P)也是公式，它表示这样一个命题：对于所有元组 t 使公式 P 为真。元组变量的自由约束性与(3)相同。

(5) 在公式中，各种运算符的优先次序为：

① 算术比较运算符；

② 量词次之；

③ 逻辑运算符最低，且¬ 的优先级高于∧和∨的优先级；

④ 加括号时，括号中运算符优先，同一括号内的运算符之优先级遵循①、②、③。

(6) 有限次地使用上述五条规则得到的公式是元组关系演算公式，其他公式都不是元组关系演算公式。

【例 3.16】 表 3.16 的(a)、(b)是关系 R 和 S，(c)～(g)分别是下面五个元组表达式的值：

R1={t|S(t)∧t[1]>2}

R2={t|R(t)∧¬ S(t)}

R3={t|(∃u)(S(t)∧R(u)∧t[3]<u[2])}

R4={t|(∀u)(R(t)∧S(u)∧t[3]>u[1])}

R5={t|(∃u)(∃v)(R(u)∧S(v)∧u[1]>v[2]∧t[1]=u[2]∧t[2]=v[3]∧t[3]=u[1])}

表 3.16 元组关系演算的例子

A	B	C
1	2	3
4	5	6
7	8	9

(a) 关系 R

A	B	C
1	2	3
3	4	6
5	6	9

(b) 关系 S

A	B	C
3	4	6
5	6	9

(c) R1

A	B	C
4	5	6
7	8	9

(d) R2

A	B	C
1	2	3

(e) R3

A	B	C
4	5	6
7	8	9

(f) R4

R. B	S. C	R. A
5	3	4
8	3	7
8	6	7
8	9	7

(g) R5

在元组关系演算的公式中,有下列三个等价的转换规则:

(1) $P_1 \wedge P_2$ 等价于$\neg(\neg P_1 \vee \neg P_2)$;$P_1 \vee P_2$ 等价于$\neg(\neg P_1 \wedge \neg P_2)$。

(2) $(\forall s)(P_1(s))$等价于$\neg(\exists s)(\neg P_1(s))$;$(\exists s)(P_1(s))$等价于$\neg(\forall s)(\neg P_1(s))$。

(3) $P_1 \Rightarrow P_2$ 等价于$\neg P_1 \vee P_2$。

2. 关系代数表达式到元组表达式的转换

可以把关系代数表达式等价地转换为元组表达式。由于所有的关系代数表达式都能用五个基本操作组合而成,因此只需把五个基本操作转换为元组演算表达式。下面举例说明。

【例 3.17】 设关系 R 和 S 都是三元关系,那么关系 R 和 S 的五个基本操作可直接转化成等价的元组关系演算表达式:

R∪S 可用$\{t|R(t) \vee S(t)\}$表示;

R−S 可用$\{t|R(t) \wedge \neg S(t)\}$表示;

R×S 可用$\{t|(\exists u)(\exists v)(R(u) \wedge S(V) \wedge t[1]=u[1] \wedge t[2]=u[2] \wedge t[3]=u[3] \wedge t[4]=v[1] \wedge t[5]=v[2] \wedge t[6]=v[3])\}$表示。

设投影操作是 $\pi_{2,3}(R)$,那么元组表达式可写成:

$$\{t \mid (\exists u)(R(u) \wedge t[1]=u[2] \wedge t[2]=u[3])\}$$

$\sigma_F(R)$可用$\{t|R(t) \wedge F'\}$表示,F'是 F 的等价表示形式。例如 $\sigma_{2='d'}(R)$可写成$\{t|(R(t) \wedge t[2]='d')\}$。

【例 3.18】 设关系 R 和 S 都是二元关系,把关系代数表达式 $\pi_{1,4}(\sigma_{2=3}(R \times S))$转换成元组表达式的过程由里向外进行,如下所述。

(1) R×S 可用$\{t|(\exists u)(\exists v)(R(u) \wedge S(v) \wedge t[1]=u[1] \wedge t[2]=u[2] \wedge t[3]=v[1] \wedge t[4]=v[2])\}$表示。

(2) 对于 $\sigma_{2=3}(R \times S)$,只要在上述表达式的公式中加上"$\wedge t[2]=t[3]$"即可。

(3) 对于 $\pi_{1,4}(\sigma_{2=3}(R \times S))$,可得到下面的元组表达式:

$$\{w \mid (\exists t)(\exists u)(\exists v)(R(u) \wedge S(v) \wedge t[1]=u[1] \wedge t[2]=u[2] \wedge t[3]=v[1] \wedge t[4]=v[2] \wedge t[2]=t[3] \wedge w[1]=t[1] \wedge w[2]=t[4])\}$$

(4) 再对上式化简,去掉元组变量 t,可得下式:

$$\{w \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge u[2]=v[1] \wedge w[1]=u[1] \wedge w[2]=v[2])\}$$

【例 3.19】 对于例 3.12 中查询语句的关系代数表达式形式也可以用元组表达式形式表示。

(1) 检索供应零件给工程 J1 的供应商编号 SNO 与零件编号 PNO。

$$\{t \mid (\exists u)(SPJ(u) \wedge u[3]='J1' \wedge t[1]=u[1] \wedge t[2]=u[2])\}$$

(2) 检索供应零件给工程 J1,且零件编号为 P1 的供应商编号 SNO。

$$\{t \mid (\exists u)(SPJ(u) \wedge u[2]='P1' \wedge u[3]='J1' \wedge t[1]=u[1])\}$$

(3) 检索使用了编号为 P3 零件的工程编号和名称。

$$\{t \mid (\exists u)(\exists v)(J(u) \wedge SPJ(v) \wedge v[2]='P3' \wedge u[1]=v[3] \wedge t[1]=u[1] \wedge t[2]=u[2])\}$$

(4) 检索供应零件给工程 J1,且零件颜色为红色的供应商名称 SNAME 和地址 SADDR。

$$\{t \mid (\exists u)(\exists v)(\exists w)(S(u) \wedge SPJ(v) \wedge P(w) \wedge u[1]=v[1] \wedge v[2]$$

$$=w[1] \wedge W[3]='红色' \wedge v[3]='J1' \wedge t[1]=u[2] \wedge t[2]=u[3])\}$$

(5) 检索使用了零件编号为 P3 或 P5 零件的工程编号 JNO。

$$\{t \mid (\exists u)(SPJ(u) \wedge (u[2]='P3' \vee u[2]='P5') \wedge t[1]=u[3])\}$$

(6) 检索至少使用了编号为 P3 和 P5 零件的工程编号 JNO。

$$\{t \mid (\exists u)(\exists v)(SPJ(u) \wedge SPJ(v) \wedge u[3]=v[3] \wedge u[2]='P3' \wedge v[2]='P5' \wedge t[1]=u[3])\}$$

(7) 检索不使用编号为 P3 零件的工程编号 JNO 和工程名称 JNAME。

$$\{t \mid (\exists u)(\forall v)(J(u) \wedge SPJ(v) \wedge (u[1]=v[3] \Rightarrow v[2] \neq 'P3') \wedge t[1]=u[1] \wedge t[2]=u[2])\}$$

(8) 检索使用了全部零件的工程名称 JNAME。

$$\{t \mid (\exists u)(\forall v)(\exists w)(J(u) \wedge P(v) \wedge SPJ(w) \wedge u[1]=w[3] \wedge v[1]=w[2] \wedge t[1]=u[2])\}$$

(9) 检索使用零件包含编号为 S1 的供应商所供应的全部零件的工程编号 JNO。

$$\{t \mid (\exists u)(SPJ(u) \wedge (\forall v)(SPJ(v) \wedge (v[1]='S1' \Rightarrow (\exists w)(SPJ(w) \wedge w[3]=u[3] \wedge w[2]=v[2]))) \wedge t[1]=u[3])\}$$

3.3.2 域关系演算

1. 域关系演算表达式

域关系演算(Domain Relational Calculus)类似元组关系演算,不同之处是用域变量代替元组变量的每一个分量,域变量的变化范围是某个值域而不是一个关系。可以像元组演算一样定义域演算的原子公式和公式。

原子公式有两种形式:

(1) $R(t_1,t_2,\cdots,t_k)$,R 是一个 k 元关系,每个 t_i 是常量或域变量;

(2) $x\theta y$,其中 x、y 是常量或域变量,但至少有一个是域变量,θ 是算术比较符。

域关系演算的公式中也可使用 $\wedge$、$\vee$、$\neg$ 和 $\Rightarrow$ 等逻辑运算符,也可用 $(\exists x)$ 和 $(\forall x)$ 形成新的公式,但变量 x 是域变量,不是元组变量。

自由域变量、约束域变量等概念和元组演算中一样,这里不再重复。

域演算表达式是形为:

$$\{t_1,t_2,\cdots,t_k \mid P(t_1,t_2,\cdots,t_k)\}$$

的表达式,其中 $P(t_1,t_2,\cdots,t_k)$ 是关于自由域变量 $t_1,t_2,\cdots,t_k$ 的公式。

【例 3.20】 表 3.17(a)、(b)、(c)是三个关系 R、S、W,(d)、(e)、(f)分别表示下面三个域表达式的值:

$$R1=\{xyz \mid R(xyz) \wedge x<5 \wedge y>3\}$$

$$R2=\{xyz \mid R(xyz) \vee (S(xyz) \wedge y=4)\}$$

$$R3=\{xyz \mid (\exists u)(\exists v)(R(zxu) \wedge w(yv) \wedge u>v)\}$$

2. 元组表达式到域表达式的转换

把元组表达式转换为域表达式很容易,转换规则如下:

(1) 对于 k 元的元组变量 t,可引入 k 个域变量 $t_1,t_2,\cdots,t_k$,在公式中 t 用 $t_1,t_2,\cdots,t_k$ 替换,元组分量 $t[i]$ 用 t_i 替换。

表 3.17　域关系演算的例子

A	B	C
1	2	3
4	5	6
7	8	9

(a) 关系 R

A	B	C
1	2	3
3	4	6
5	6	9

(b) 关系 S

D	E
7	5
4	8

(c) 关系 W

A	B	C
4	5	6

(d) R1

A	B	C
1	2	3
4	5	6
7	8	9
3	4	6

(e) R2

B	D	A
5	7	4
8	7	7
8	4	7

(f) R3

(2) 对于每个量词(∃u)或(∀u),若 u 是 m 元的元组变量,则引入 m 个新的域变量 u_1,u_2,…,u_m。在量词的辖域内,u 用 u_1,u_2,…,u_m 替换,u[i]用 u_i 替换,(∃u)用(∃u_1),(∃u_2),…,(∃u_m)替换,(∀u)用(∀u_1),(∀u_2),…,(∀u_m)替换。

【例 3.21】 对于例 3.18 转换成的元组表达式:

$\{w \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge u[2]=v[1] \wedge w[1]=u[1] \wedge w[2]=v[2])\}$

可用上述转换方法转换成域表达式:

$\{w_1 w_2 \mid (\exists u_1)(\exists u_2)(\exists v_1)(\exists v_2)(R(u_1 u_2) \wedge S(v_1 v_2) \wedge u_2 = v_1 \wedge w_1 = u_1 \wedge w_2 = v_2)\}$

再进一步简化,可消去域变量 u_1、v_1、v_2,得到下式:

$$\{w_1 w_2 \mid (\exists u_2)(R(w_1 u_2) \wedge S(u_2 w_2))\}$$

【例 3.22】 对于例 3.19 的查询,可转换成下列域表达式:

(1) 检索供应零件给工程 J1 的供应商编号 SNO 与零件编号 PNO。

$\{t_1 t_2 \mid (\exists u_1)(\exists u_2)(\exists u_3)(\exists u_4)(\exists u_5)(SPJ(u_1 u_2 u_3 u_4 u_5) \wedge u_3 = \text{'J1'} \wedge t_1 = u_1 \wedge t_2 = u_2)\}$

进而可简化为:

$$\{t_1 t_2 \mid (\exists u_4)(\exists u_5)(SPJ(t_1 t_2 \text{'J1'} u_4 u_5))\}$$

(2) 检索供应零件给工程 J1,且零件编号为 P1 的供应商编号 SNO。

$$\{t_1 \mid (\exists u_4)(\exists u_5)(SPJ(t_1 \text{'P1''J1'} u_4 u_5))\}$$

(3) 检索使用了编号为 P3 零件的工程编号和名称。

$\{t_1 t_2 \mid (\exists u_1)(\exists u_2)(\exists u_3)(\exists u_4)(\exists v_1)(\exists v_2)(\exists v_3)(\exists v_4)(\exists v_5)(J(u_1 u_2 u_3 u_4)$
$\wedge SPJ(v_1 v_2 v_3 v_4 v_5) \wedge v_2 = \text{'P3'} \wedge u_1 = v_3 \wedge t_1 = u_1 \wedge t_2 = u_2)\}$

可简化为:

$$\{t_1 t_2 \mid (\exists u_3)(\exists u_4)(\exists v_1)(\exists v_4)(\exists v_5)(J(t_1 t_2 u_3 u_4) \wedge SPJ(v_1 \text{'P3'} t_1 v_4 v_5))\}$$

读者可以自己写出其他查询语句的域表达式。

3.3.3　关系运算的安全性和等价性

1. 关系运算的安全性

从关系代数操作的定义可以看出,任何一个有限关系上的关系代数操作结果都不会导

致无限关系和无穷验证，所以关系代数系统总是安全的。然而，元组关系演算系统和域关系演算系统可能产生无限关系和无穷验证。例如：{t|¬ R(t)}表示所有不在关系 R 中的元组的集合，是一个无限关系。无限关系的演算需要具有无限存储容量的计算机；另外若判断公式(∃u)(w(u))和(∀u)(w(u))的真和假，需对所有的元组 u 验证，即要求进行无限次验证。显然这是毫无意义的。因此对元组关系演算要进行安全约束。安全约束是对关系演算表达式施加限制条件，对表达式中的变量取值规定一个范围，使之不产生无限关系和无穷次验证，这种表达式被称为是安全表达式。在关系演算中，约定运算只对表达式中公式涉及的关系值范围内的变量进行操作，这样就不会产生无限关系和无穷次验证问题，关系演算才是安全的。

2. 关系运算的等价性

并、差、笛卡儿积、投影和选择是关系代数最基本的操作，并构成了关系代数运算的最小完备集。已经证明，在这个基础上，关系代数、安全的元组关系演算、安全的域关系演算在关系的表达和操作能力上是等价的。

关系运算主要有关系代数、元组演算和域演算三种，典型的关系查询语言也已研制出来，它们典型的代表是 ISBL、QUEL 和 QBE 语言。

ISBL(Information System Base Language)是 IBM 公司英格兰底特律科学中心在 1976 年研制出来的，用在一个实验系统 PRTV(Peterlee Relational Test Vehicle)上。ISBL 语言与关系代数非常接近，每个查询语句都近似于一个关系代数表达式。

QUEL(Query Language)是美国伯克利加州大学研制的关系数据库系统 INGRES 的查询语言，1975 年投入运行，并由美国关系技术公司制成商品推向市场。QUEL 是一种基于元组关系演算的并具有完整的数据定义、检索、更新等功能的数据语言。

QBE(Query By Example，按例查询)是一种特殊的屏幕编辑语言。QBE 是 M. M. Zloof 提出的，在约克镇 IBM 高级研究实验室为图形显示终端用户设计的一种域演算语言。1978 年在 IBM370 上实现。QBE 使用起来很方便，属于人机交互语言，用户可以是没有计算机知识和数学基础的非程序用户。现在，QBE 的思想已渗入许多 DBMS 中。

还有一个语言 SQL，这是介乎关系代数和元组演算之间的一种关系查询语言，现已成为关系数据库的标准语言，将在第 4 章详细介绍。

3.4 查 询 优 化

3.4.1 关系代数表达式的优化问题

查询处理的代价通常取决于磁盘访问，磁盘访问比内存访问速度要慢得多。对于一个给定的查询，通常会有很多可能的处理策略，也就是可以写出许多等价的关系代数表达式。就所需磁盘访问次数而言，策略好坏差别很大，有时甚至相差几个数量级。因此，系统多花点时间在选择一个较好的查询处理策略上是值得的，即便该查询语句只执行一次。

在关系代数表达式中需要指出若干关系的操作步骤。那么，系统应该以什么样的操作顺序，才能做到既省时间，又省空间，而且效率也比较高呢？这个问题称为查询优化问题。

在关系代数运算中,笛卡儿积和连接运算是最费时间的。若关系 R 有 m 个元组,关系 S 有 n 个元组,那么 R×S 就有 $m \times n$ 个元组。当关系很大时,R 和 S 本身就要占较大的外存空间,由于内存的容量是有限的,只能把 R 和 S 的一部分元组读进内存,如何有效地执行笛卡儿积操作,花费较少的时间和空间,就有一个查询优化的策略问题。

【例 3.23】 设关系 R 和 S 都是二元关系,属性名分别为 A、B 和 C、D。设有一个查询可用关系代数表达式表示:

$$E1=\pi_A(\sigma_{B=C \wedge D='99'}(R \times S))$$

也可以把选择条件 D='99'移到笛卡儿积中的关系 S 前面:

$$E2=\pi_A(\sigma_{B=C}(R \times \sigma_{D='99'}(S)))$$

还可以把选择条件 B=C 与笛卡儿积结合成等值连接形式:

$$E3=\pi_A(R \underset{B=C}{\bowtie} \sigma_{D='99'}(S))$$

这三个关系代数表达式是等价的,但执行的效率大不一样。显然,求 E1、E2、E3 的大部分时间是花在连接操作上。

对于 E1,先做笛卡儿积,要把 R 的每个元组与 S 的每个元组连接起来。在外存储器中,每个关系以文件形式存储。设关系 R 和 S 的元组个数都是 10000,每个物理存储块可存放 5 个元组,那么关系 R 有 2000 块,S 也有 2000 块。而内存只给这个操作 100 块的内存空间。此时执行笛卡儿积操作较好的方法是先让 R 的第一组 99 块数据装入内存,然后关系 S 逐块转入内存去做元组的连接;再把关系 R 的第二组 99 块数据装入内存,然后关系 S 逐块转入内存去做元组的连接。

这样关系 R 每块只装入内存一次,装入块数是 2000;而关系 S 的每块需要装入内存(2000/99)次,装入内存的块数是(2000/99)×2000,因而执行 R×S 的总装入块数是:

$$2000+(2000/99)\times 2000 \approx 42\,400(\text{块})$$

若每秒装入内存 20 块,则需要约 35 分钟。这里还没有考虑连接后产生的元组写入外存的时间。

对于 E2 和 E3,由于先做选择,设 S 中 D='99'的元组只有几个,因此关系的每块只需装入内存一次,则关系 R 和 S 的总装入块数为 4000,约 3 分钟,相当于求 E1 花费时间的 1/10。

如果对关系 R 和 S 在属性 B、C、D 上建立索引,那么花费时间还要少得多。

这种差别的原因是计算 E1 时 S 的每个元组装入内存多次,而计算 E2 和 E3 时,S 的每个元组只进内存一次。在计算 E3 时把笛卡儿积和选择操作合并成等值连接操作。

从此例可以看出,如何安排选择、投影和连接的顺序是一个很重要的问题。

3.4.2 关系代数表达式的等价变换规则

两个关系代数表达式等价是指用同样的关系实例代替两个表达式中相应关系时所得到的结果是一样的。也就是得到相同的属性集和相同的元组集,但元组中属性的顺序可能不一致。两个关系代数表达式 E1 和 E2 的等价写成 E1≡E2。

涉及连接和笛卡儿积的等价变换规则有下面两条。

1. 连接和笛卡儿积的交换律

设 E1 和 E2 是关系代数表达式,F 是连接的条件,那么下列式子成立(不考虑属性间的

顺序)：

$$E1 \underset{F}{\bowtie} E2 \equiv E2 \underset{F}{\bowtie} E1$$

$$E1 \bowtie E2 \equiv E2 \bowtie E1$$

$$E1 \times E2 \equiv E2 \times E1$$

2. 连接和笛卡儿积的结合律

设 E1、E2 和 E3 是关系代数表达式，F1 和 F2 是连接条件，F1 只涉及 E1 和 E2 的属性，F2 只涉及 E2 和 E3 的属性，那么下列式子成立：

$$(E1 \underset{F1}{\bowtie} E2) \underset{F2}{\bowtie} E3 \equiv E1 \underset{F1}{\bowtie} (E2 \underset{F2}{\bowtie} E3)$$

$$(E1 \bowtie E2) \bowtie E3 \equiv E1 \bowtie (E2 \bowtie E3)$$

$$(E1 \times E2) \times E3 \equiv E1 \times (E2 \times E3)$$

涉及选择的规则有下面的规则 3～12。

3. 投影的串接

设 L1，L2，…，Ln 为属性集，并且 L1⊆L2⊆…⊆Ln，那么下式成立：

$$\pi_{L1}(\pi_{L2}(\cdots(\pi_{Ln}(E))\cdots)) \equiv \pi_{L1}(E)$$

4. 选择的串接

$$\sigma_{F1}(\sigma_{F2}(E)) \equiv \sigma_{F1 \wedge F2}(E)$$

由于 $F1 \wedge F2 = F2 \wedge F1$，因此选择的交换律也成立：

$$\sigma_{F1}(\sigma_{F1}(E)) \equiv \sigma_{F2}(\sigma_{F1}(E))$$

5. 选择和投影操作的交换

$$\pi_{L}(\sigma_{F}(E)) \equiv \sigma_{F}(\pi_{L}(E))$$

这里要求 F 只涉及 L 中的属性，如果条件 F 还涉及不在 L 中的属性集 L1，那么就有下式成立：

$$\pi_{L}(\sigma_{F}(E)) \equiv \pi_{L}(\sigma_{F}(\pi_{L \cup L1}(E)))$$

6. 选择对笛卡儿积的分配律

$$\sigma_{F}(E1 \times E2) \equiv \sigma_{F}(E1) \times E2$$

这里要求 F 只涉及 E1 中的属性。

如果 F 形为 $F1 \wedge F2$，且 F1 只涉及 E1 的属性，F2 只涉及 E2 的属性，那么使用规则 4 和 6 可得到下列式子：

$$\sigma_{F}(E1 \times E2) \equiv \sigma_{F1}(E1) \times \sigma_{F2}(E2)$$

此外，如果 F 形为 $F1 \wedge F2$，且 F1 只涉及 E1 的属性，F2 只涉及 E1 和 E2 的属性，那么可得到下列式子：

$$\sigma_{F}(E1 \times E2) \equiv \sigma_{F2}(\sigma_{F1}(E1) \times E2)$$

也就是把一部分选择条件放到笛卡儿积中关系的前面。

7. 选择对并的分配律

$$\sigma_{F}(E1 \cup E2) \equiv \sigma_{F1}(E1) \cup \sigma_{F2}(E2)$$

这里要求 E1 和 E2 具有相同的属性名，或者 E1 和 E2 表达的关系的属性有对应性。

8. 选择对集合差的分配律

$$\sigma_{F}(E1 - E2) \equiv \sigma_{F}(E1) - \sigma_{F}(E2)$$

或

$$\sigma_F(E1-E2)\equiv\sigma_F(E1)-E2$$

这里也要求 E1 和 E2 的属性有对应性。恒等式右边的 $\sigma_F(E2)$也可以不做选择操作，直接用 E2 代替，但往往求 $\sigma_F(E2)$比求 E2 容易。

9. 选择对自然连接的分配律

如果 F 只涉及表达式 E1 和 E2 的公共属性，那么选择对自然连接的分配律成立：

$$\sigma_F(E1\bowtie E2)\equiv\sigma_F(E1)\bowtie\sigma_F(E2)$$

10. 投影对笛卡儿积的分配律

$$\pi_{L1\cup L2}(E1\times E2)\equiv\pi_{L1}(E1)\times\pi_{L2}(E2)$$

这里要求 L1 是 E1 中的属性集，L2 是 E2 中的属性集。

11. 投影对并的分配律

$$\pi_L(E1\cup E2)\equiv\pi_L(E1)\cup\pi_L(E2)$$

这里要求 E1 和 E2 的属性有对应性。

12. 选择与连接操作的结合

根据 F 连接的定义可得

$$\sigma_F(E1\times E2)\equiv E1\underset{F}{\bowtie}E2$$

$$\sigma_{F1}(E1\underset{F2}{\bowtie}E2)\equiv E1\underset{F2\wedge F2}{\bowtie}E2$$

涉及集合操作的有下面两条规则。

13. 并和交的交换律

$$E1\cup E2\equiv E2\cup E1$$

$$E1\cap E2\equiv E2\cap E1$$

14. 并和交的结合律

$$(E1\cup E2)\cup E3\equiv E1\cup(E2\cup E3)$$

$$(E1\cap E2)\cap E3\equiv E1\cap(E2\cap E3)$$

3.4.3 优化的一般策略

这里介绍的优化策略与关系的存储技术无关，主要是如何安排操作的顺序。但经过优化后的表达式不一定是所有等价表达式中执行时间最少的。此处不讨论执行时间最少的“最优问题”，只是介绍优化的一般技术。主要有以下一些策略。

(1) 在关系代数表达式中尽可能早地执行选择操作。对于有选择运算的表达式，应尽量提前执行选择操作，以得到较小的中间结果，减少运算量和读外存块的次数。

(2) 把笛卡儿积和其后的选择操作合并成 F 连接运算。因为两个关系的笛卡儿积是一个元组数较大的关系(中间结果)，而做了选择操作后，可能会获得很小的关系。这两个操作一起做，即对每一个连接后的元组，立即检查是否满足选择决定条件，再决定取舍，将会减少时间和空间的开销。

(3) 同时计算一连串的选择和投影操作，以免分开运算造成多次扫描文件，从而能节省操作时间。

因为选择和投影都是一元操作符，它们把关系中的元组看成是独立的单位，所以可以对

每个元组连续做一串操作(当然顺序不能随意改动)。如果在一个二元运算后面跟着一串一元运算,那么也可以结合起来同时操作。

(4) 如果在一个表达式中多次出现某个子表达式,那么应该将该子表达式预先计算出结果保存起来,以免重复计算。

(5) 适当地对关系文件进行预处理。关系以文件形式存储,根据实际需要对文件进行排序或建立索引文件,这样能使两个关系在进行连接时,可以很快地、有效地对应起来。有时,建立永久的排序文件和索引文件需要占据大量空间,因此也可临时产生文件,这只是花些时间,但还是合算的。

(6) 在计算表达式前应先估计一下怎么计算合算。例如,计算 R×S,应先查看一下 R 和 S 的物理块数,然后再决定哪个关系可以只进内存一次,而另一关系进内存多次,这样才合算。

3.4.4 优化算法

关系代数表达式的优化是由 DBMS 的 DML 编译器完成的。对一个关系代数表达式进行语法分析,可以得到一棵语法树,叶子是关系,非叶子节点是关系代数操作。利用前面的等价变换规则和优化策略来对关系代数表达式进行优化。

算法 3.1 关系代数表达式的优化。

输入:一个关系代数表达式的语法树。

输出:计算表达式的一个优化序列。

方法:依次执行下面每一步。

(1) 使用等价变换规则 4 把每个形为 $\sigma_{F1\wedge\cdots\wedge Fn}(E)$ 的子表达式转换成选择串接形式:

$$\sigma_{F1}(\cdots\sigma_{Fn}(E)\cdots)$$

(2) 对每个选择操作,使用规则 4～9,尽可能地把选择操作移近树的叶端(即尽可能早地执行选择操作)。

(3) 对每个投影操作,使用规则 3、5、10、11,尽可能把投影操作移近树的叶端。规则 3 可能使某些投影操作消失,而规则 5 可能会把一个投影分成两个投影操作,其中一个将靠近叶端。如果一个投影是针对被投影的表达式的全部属性,则可消去该投影操作。

(4) 使用规则 3～5,把选择和投影合并成单个选择、单个投影或一个选择后跟一个投影。使多个选择、投影能同时执行或在一次扫描中同时完成。

(5) 将上述步骤得到的语法树的内节点分组。每个二元运算(×、∪、－)节点与其直接祖先(不超过别的二元运算节点)的一元运算节点(σ 或 π)分为一组。如果它的子孙节点一直到叶都是一元运算符(σ 或 π),则也并入该组。但是,如果二元运算是笛卡儿积,而且后面不是与它组合成等值连接的选择时,则不能将选择与这个二元运算组成同一组。

(6) 生成一个程序,每一组节点的计算是程序中的一步,各步的顺序是任意的只要保证任何一组不会在它的子孙组之前计算。

【例 3.24】 对于工程项目数据库。

供应商关系　S(SNO, SNAME, SADDR)

零件关系　P(PNO, PNAME, COLOR, WEIGHT)

工程关系　J(JNO, JNAME, JCITY, BALANCE)

供应关系　　SPJ(SNO, PNO, JNO, PRICE, QTY)

现有一个查询语句：检索供应给工程 J1 零件为红色的供应商名称和地址。

$$\pi_{SNAME,SADDR}(\sigma_{JNO='J1'\wedge COLOR='红色'}(S \bowtie SPJ \bowtie P))$$

对于上述式子中的⋈符号用 π、σ、×操作表示，可得下式：

$$\pi_{SNAME,SADDR}(\sigma_{JNO='J1'\wedge COLOR='红色'}(\pi_{L}(\sigma_{S.SNO=SPJ.SNO\wedge SPJ.PNO=P.PNO}(S\times SPJ\times P))))$$

此处 L 是 SNO, SNAME, SADDR, PNO, JNO, PRICE, QTY, PNAME, COLOR, WEIGHT。关系代数表达式构成的语法树如图 3.5 所示，下面使用优化算法对语法树进行优化。

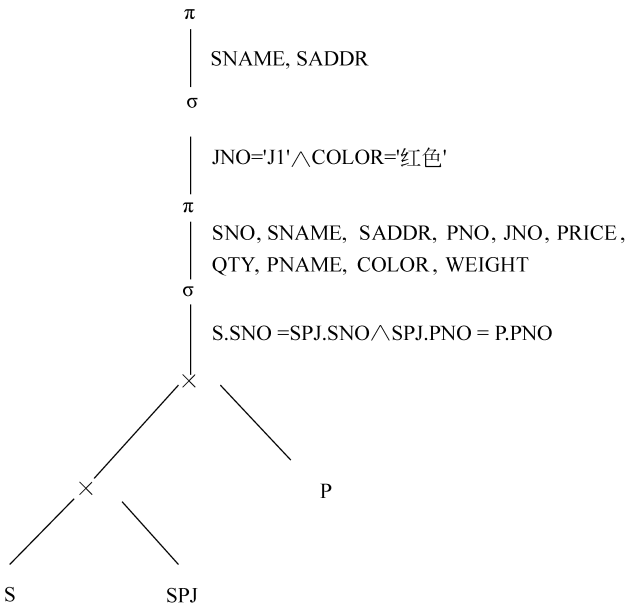

图 3.5　关系代数表达式构成的语法树

(1) 将每个选择运算分裂成两个选择运算，共得到四个选择操作。

$$\sigma_{JNO='J1'}$$

$$\sigma_{COLOR='红色'}$$

$$\sigma_{S.SNO=SPJ.SNO}$$

$$\sigma_{SPJ.PNO=P.PNO}$$

(2) 使用等价变换规则 4～8，把四个选择运算尽可能地向树的叶端靠拢。据规则 4 和 5，可以把 $\sigma_{JNO='J1'}$ 和 $\sigma_{COLOR='红色'}$ 移到投影和另两个选择操作下面，直接放在笛卡儿积外面得到子表达式：

$$\sigma_{COLOR='红色'}(\sigma_{JNO='J1'}(S\times SPJ)\times P)$$

其中，内层选择仅涉及关系 SPJ，外层选择仅涉及关系 P，所以上式可变换成：

$$(\sigma_{JNO='J1'}(SPJ)\times S)\times\sigma_{COLOR='红色'}(P)$$

$\sigma_{SPJ.PNO=P.PNO}$ 不能再往叶端移动了，因为它的属性涉及两个关系 SPJ 和 S，但 $\sigma_{S.SNO=SPJ.SNO}$ 还可向下移，与笛卡儿积交换位置。

然后根据规则 3，再把两个投影合并成一个投影 $\pi_{SNO,SNAME}$。这样，原来的语法树

(图 3.5)变成了如图 3.6 所示的形式。

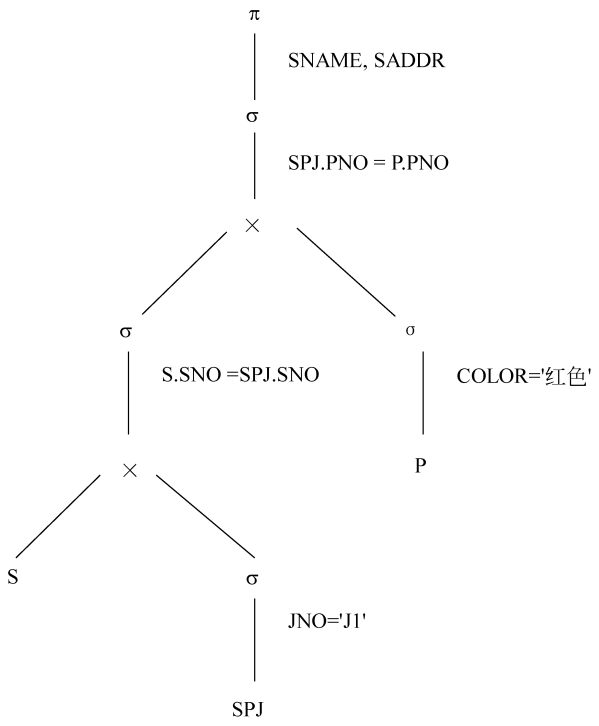

图 3.6　优化过程中的语法树

(3) 据规则 5,把投影和选择进行交换,在 σ 前增加一个 π 操作。用

$$\pi_{SNAME,SADDR}$$
$$\sigma_{SPJ.PNO=P.PNO}$$
$$\pi_{SNAME,SADDR,SPJ.PNO,P.PNO}$$

代替 $\pi_{SNAME,SADDR}$ 和 $\sigma_{SPJ.PNO=P.PNO}$。

再把 $\pi_{SNAME,SADDR,SPJ.PNO,P.PNO}$ 分成 $\pi_{SNAME,SADDR,SPJ.PNO}$ 和 $\pi_{P.PNO}$,使它们分别对 $\sigma_{S.SNO=SPJ.SNO}(\cdots)$和 $\sigma_{COLOR='红色'}(P)$做投影操作。

再据规则 5,将投影 $\pi_{SNAME,SADDR,SPJ.PNO}$ 和 $\pi_{P.PNO}$ 分别与前面的选择运算形成两个串接运算:

$$\pi_{SNAME,SADDR,SPJ.PNO}$$
$$\sigma_{S.SNO=SPJ.SNO}$$
$$\pi_{S.SNO,SNAME,SADDR,SPJ.SNO,SPJ.PNO}$$

$$\pi_{P.PNO}$$
$$\sigma_{COLOR='红色'}$$

再把 $\pi_{S.SNO,SNAME,SADDR,SPJ.PNO}$ 往叶端移,形成图 3.7 的语法树。图 3.7 中用虚线划分了两个运算组。

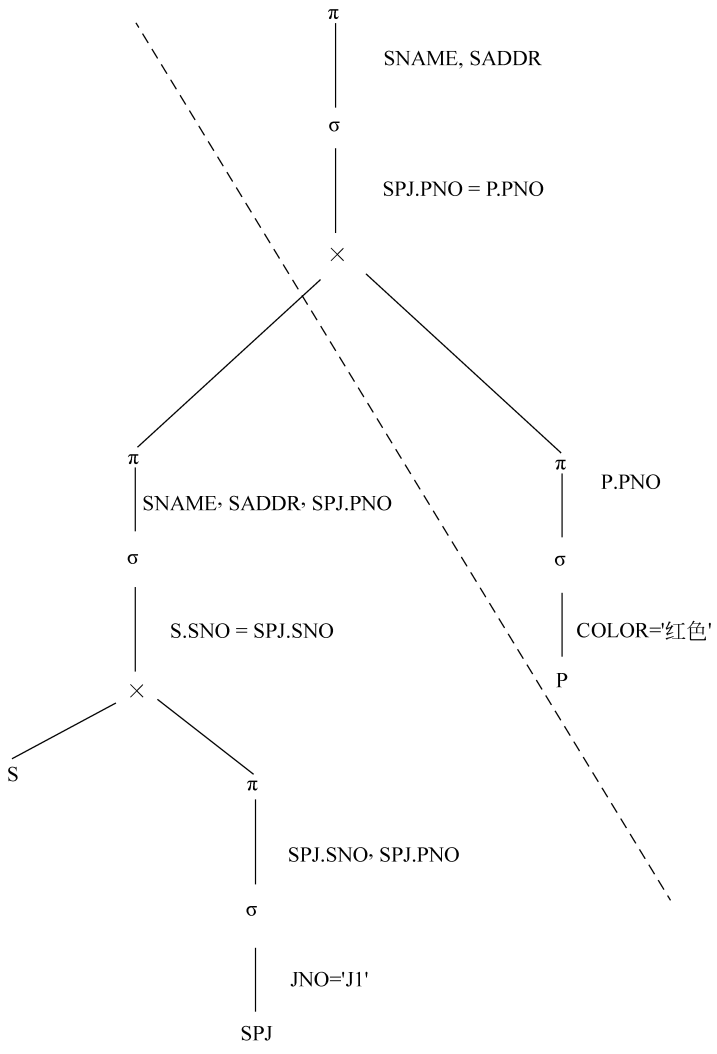

图 3.7　优化的语法树及其分组

(4) 执行时从叶端依次向上进行,每组运算只对关系扫描一次。

小　　结

关系运算理论是关系数据库查询语言的理论基础。只有掌握了关系运算理论,才能深刻理解查询语言的本质和熟练使用查询语言。

本章先介绍了关系模型的基本概念。关系定义为元组的集合,但关系又有它特殊的性质。关系模型必须遵循实体完整性规则、参照完整性规则和用户定义的完整性规则。

关系代数的五个基本操作(构成一个完备集)以及四个组合操作,是本章的重点。要能进行两方面的运用:一是计算关系代数表达式的值;二是根据查询语句写出关系代数表达式的表示形式。

关系演算是基于谓词演算的关系运算,理论性较强。主要理解表达式的语义,计算其

值，并能根据简单的查询语句写出元组表达式。

查询优化是指系统对关系代数表达式要进行优化组合，以提高系统效率。本章介绍了关系代数表达式的若干变换规则和优化的一般策略，然后提出了一个查询优化的算法。

习　题　3

1. 名词解释。

关系模型	关系模式	关系实例	属性
域	元组	超键	候选键
主键	外键	实体完整性规则	参照完整性规则
过程性语言	非过程性语言	无限关系	无穷验证

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元组？

4. 关系与普通表格、文件有什么区别？

5. 笛卡儿积、等值连接、自然连接三者之间有什么区别？

6. 设有关系 R 和 S(见表 3.18)，计算 $R \cup S$，$R - S$，$R \cap S$，$R \times S$，$\pi_{3,2}(S)$，$\sigma_{B<'5'}(R)$，$R \underset{2<2}{\bowtie} S$，$R \bowtie S$。

表 3.18　习题 3.6

R：

A	B	C
3	6	7
2	5	7
7	2	3
4	4	3

S：

A	B	C
3	4	5
7	2	3

7. 如果 R 是二元关系，那么下列元组表达式的结果是什么？

$$\{t \mid (\exists u)(R(t) \wedge R(u) \wedge (t[1] \neq u[1] \vee t[2] \neq u[2]))\}$$

8. 假设 R 和 S 分别是三元和二元关系，试把表达式 $\pi_{1,5}(\sigma_{2=4 \vee 3=4}(R \times S))$ 转换成等价的。

(1) 汉语查询句子；(2) 元组表达式；(3) 域表达式。

9. 假设 R 和 S 都是二元关系，试把元组表达式 $\{t \mid R(t) \wedge (\exists u)(S(u) \wedge u[1] \neq t[2])\}$ 转换成等价的。

(1) 汉语查询句子；(2) 域表达式；(3) 关系代数表达式。

10. 试把域表达式 $\{ab \mid R(ab) \wedge R(ba)\}$ 转换成等价的。

(1) 汉语查询句子；(2) 关系代数表达式；(3) 元组表达式。

11. 有两个关系 R(A，B，C)和 S(D，E，F)，试把下列关系代数表达式转换成等价的元组表达式。

(1) $\pi_A(R)$；(2) $\sigma_{B='17'}(R)$；(3) $R \times S$；(4) $\pi_{A,F}(\sigma_{C=D}(R \times S))$。

12. 设有三个关系：

```
S(SNO, SNAME, AGE, SEX,SDEPT)
SC(SNO, CNO, GRADE)
C(CNO, CNAME,CDEPT, TNAME)
```

试用关系代数表达式表示下列查询语句。

(1) 检索 LIU 老师所授课程的课程号、课程名。

(2) 检索年龄大于 23 岁的男学生的学号与姓名。

(3) 检索学号为 S3 学生所学课程的课程名与任课教师名。

(4) 检索至少选修 LIU 老师所授课程中一门课的女学生姓名。

(5) 检索 WAN 同学不学的课程的课程号。

(6) 检索至少选修两门课程的学生学号。

(7) 检索全部学生都选修的课程的课程号与课程名。

(8) 检索选修课程包含 LIU 老师所授课程的学生学号。

13. 试用元组表达式表示习题 12 题的各个查询语句。

14. 试用域表达式表示习题 12 题的各个查询语句。

15. 在习题 12 题的三个关系中,用户有一查询语句：检索数学系的学生选修计算机课程的课程名和任课教师姓名。

(1) 试写出该查询的关系代数表达式。

(2) 画出该查询初始的关系代数表达式的语法树。

(3) 使用 3.4.4 节优化算法,对语法树进行优化,试写出该查询优化的关系代数表达式。

(4) 画出优化后的语法树。

16. 为什么要对关系代数表达式进行优化?