

数据库系统原理

教程：数据库系统概论（第5版）

结合：CMU 15-445/645 INTRO TO DATABASE SYSTEMS

华中科技大学 计算机学院

左琼

2.4 关系代数运算小结

- 5 种基本运算 (并 \cup 、差 $-$ 、笛卡尔积 \times 、选择 σ 、投影 π)
- 其它运算 (交 \cap 、连接 \bowtie 、除 \div) 均可用 5 种基本运算来表达, 引进它们并不增加语言的能力, 但可以简化表达:
 - $R \cap S = R - (R - S) = S - (S - R)$
 - $R \bowtie S = \pi_{i_1, \dots, i_m}(\sigma_{R.A_1=S.A_1 \wedge R.A_2=S.A_2 \dots \wedge R.A_k=S.A_k}(R \times S))$
 - $R \div S = \pi_{1,2,\dots,r-s}(R) - \pi_{1,2,\dots,r-s}((\pi_{1,2,\dots,r-s}(R) \times S) - R)$
- 关系代数中, 这些运算经有限次复合后形成的式子称为**关系代数表达式**。利用这些表达式可以实现对关系数据库的各种操作 (插入、删除、修改、查询)。

2.4.3 附加运算* (了解*)

□ 问题 关系代数的基本运算足以表达任何查询，但使用不方便，写出的表达式太长。定义一些附加运算，不能增加关系代数功能，但能简化表达式。

□ 更名运算

$$\rho_{x(A_1, A_2, \dots, A_n)}(E)$$

将E的结果取名字x，且将属性名改为A1,...An

□ 为什么需要更名运算？

一个关系代数表达式的结果关系没有名字；

一个关系表可以多次参与到一个联系中，每次参与角色不同，如何区分？

更名运算示例*

- 例：只用基本关系运算符，找出学生中最大年龄？
- 求解：利用集合的补集运算，求出所有非最大年龄集合的补集即可，反证思维
 - 1) 求出一个由非最大年龄构成的表
 - 2) 求所有学生年龄与上一步结果的差

如何求非最大年龄集合？

令所有学生集合为A，所有年龄集合为B，显然，A与B都在Student表中；

对于A中每个学生t，将其与B中所有年龄元组配对，判断：该学生年龄是否小于所有学生年龄集中的某一个。若是，则将其放在结果集中。

更名运算示例*

分析知，**Student表两次以不同角色参与运算**，为区别起见，令代表年龄的学生表为d。

S1: 首先将学生表与d表合并

$$\text{Student} \times \rho_d(\text{Student})$$

S2: 在合并表中选择哪些年龄比所有年龄中某一个小的学生

$$\pi_{\text{Student.age}}(\sigma_{\text{Student.age} < d.\text{age}}(\text{Student} \times \rho_d(\text{Student})))$$

S3: 求差 (求问题的反面)

$$\pi_{\text{age}}(\text{Student}) - \pi_{\text{Student.age}}(\sigma_{\text{Student.age} < d.\text{age}}(\text{Student} \times \rho_d(\text{Student})))$$

注: 上面S1、S2步中，也可以用条件连接的选择功能，直接从学生表中将哪些年龄不是最大的学生选择出

$$\rho_s(\text{Student}) \bowtie_{s.\text{age} < d.\text{age}} \rho_d(\text{Student})$$

思考题: 求出与“张三”在同一个系的学生?

$$\rho_s(\text{Student}) \bowtie_{\sigma_{d.\text{sname} = \text{'张三'}}} \rho_d(\text{Student})$$

$$s.\text{dept} = d.\text{dept}$$

聚集运算*

- 问题示例 选课表每个学生会有多个选课记录，若要查询每个学生所选课程的平均分如何办？
- 解决办法 1) 将选课表按照学号分组
2) 对分组后的表中每个组再调用AVG () 函数

SNO	CNO	GRADE
200215121	(1,	92)
	(2,	85)
	(3,	88)
200215122	(2,	90)
	(3,	80)
.....	

聚集运算*

□ 聚集运算 将表按照属性分组，即将元组按照属性值重新组合

□ 定义

$A_1, \dots, A_n G(E)$

E是关系代数表达式，即是一张表

A_1, \dots, A_n 是用于分组的一组属性

运算符G表示将表达式E按照 A_1, \dots, A_n 分组

同一组中所有元组在 A_1, \dots, A_n 上值相同

不同组元组在 A_1, \dots, A_n 上值不同

□ 示例

$sno G(SC)$ 结果为上一页中表

$ssex G(Student \bowtie SC)$ 的结果为多少？

□ 问题

聚集运算的结果是由多个分组组成的表，对每个分组可以定义聚集函数

聚集函数*

- 聚集函数 输入为集合，输出为单一值的函数。

`sum()`, `avg()`, `max()`, `min()`, `count()`

- 应用 聚集函数往往和聚集运算组合使用

对于聚集运算后的结果，对每个组再运用聚集函数处理

- 示例 `sno Gcount (cno) ,min(grade)(SC)`

表示每个学生所选课的人数和最低分

- 如何去除相同元素？

`distinct`操作符，其含义是消除相同元素。

e.g `sno Gcount (distinct cno) ,min(grade)(SC)`

广义投影*

- 问题示例 学生表中只有年龄，若查询学生的出生年份如何办？
- 问题分析 出生年份可以通过年龄计算求得，因此，扩展投影运算，使投影属性可以是派生属性，即可以从表中经过运算得出。
- 广义投影 $\pi_{F_1, \dots, F_n}(E)$
F1,...Fn中是可以涉及常量、系统函数及E中属性的算术表达式。
- 示例 $\pi_{\text{sno}, \text{year}() - \text{age}}(\text{Student})$ 结果为由 (sno, year()-age) 两列组成的表；
 $\rho_{\text{sno}, \text{birthday}}(\pi_{\text{sno}, \text{year}() - \text{age}}(\text{Student}))$ 结果为 (sno, birthday) 两列组成的表。

数据库修改*

□ 定义 对数据库的增、删、查的运算符，通过赋值完成。

□ 删除 $r \leftarrow r - E$ 其中：E是关系代数查询表达式。

例： $SC \leftarrow SC - \sigma_{sno=2}(SC)$

表示从选课表中删除了2号学生的选课记录。

□ 插入 $r \leftarrow r \cup E$

□ 更新 $r \leftarrow \pi_{F_1, \dots, F_n}(r)$ ，使用广义投影可以改变元组中的值。

$r \leftarrow \pi_{F_1, \dots, F_n}(\sigma_P(r)) \cup (r - \sigma_P(r))$ ，对r中的部分元组修改。

□ 注 上述定义的所有运算符来自于应用语义需求，每个运算符在后面的SQL语言中都有相应语句。

视图操作*

□ 视图操作 能够从已有的表集合中生成一个虚关系表的运算。

出于安全性或出于方便性考虑，需要视图操作。

□ 定义 `creat view v as E`, 将E的结果作为视图v

□ 示例 `creat view cs-student`

`as ($\sigma_{sdept='cs'}(SC)$)`

□ 注意 视图定义运算不同于关系赋值运算：
执行赋值运算时，结果关系被计算并存储，
执行视图定义运算时，结果关系未被计算，
直到某个查询使用视图时才动态计算视图表。

关系代数随堂练习

考虑下面关系数据库：公司 com (cno, cname, city)

员工 emp (eno, ename, street, city, mno)

;mno 是外键，引用emp(eno)，表示该员工的直属上级经理

工作 works (eno, cno, salary)

请用关系代数表达式表示下面查询？

- 1) 找出公司（编号cno为1001）的所有员工姓名。
- 2) 找出公司名为“DM”的所有收入在10000元以上员工的姓名和居住城市。
- 3) 找出所有居住地与工作的公司在同一个城市的员工姓名。
- 4) 找出与其直属上级经理居住在同一个城市的所有员工姓名和他的经理姓名。
- 5) 找出“DM”公司中不在其分公司（cno为1001）工作的所有员工编号和姓名。
- 6) 找出比公司（cno为1002）的所有员工收入都高的公司名和员工姓名。
- 7) 假设公司可以在几个城市部署分部(cname相同，但cno不同)。找出在“DM”公司所在的各个城市都有分部的公司。

第二章 关系数据库

2.1 关系模型概述

2.2 关系数据结构

2.3 关系的完整性

2.4 关系代数

2.5 关系演算 *

2.6 小结

2.5 关系演算（掌握与关系代数的差异*）

□ 关系演算：以数理逻辑中的谓词演算为基础。

□ 按谓词变元不同进行分类：

1. 元组关系演算：

以元组变量作为谓词变元的基本对象

元组关系演算语言ALPHA

2. 域关系演算：

以域变量作为谓词变元的基本对象

域关系演算语言QBE

关系代数——将关系作为运算单位(操作数)，用关系代数表达式表示的运算方法。

2.5.1 元组关系演算*

- 用元组作为谓词变量的一种谓词演算方法。元组关系演算表达式的一般形式为：

$$\{t \mid P(t)\}$$

表示所有使 $P(t)$ 为真的元组集合。

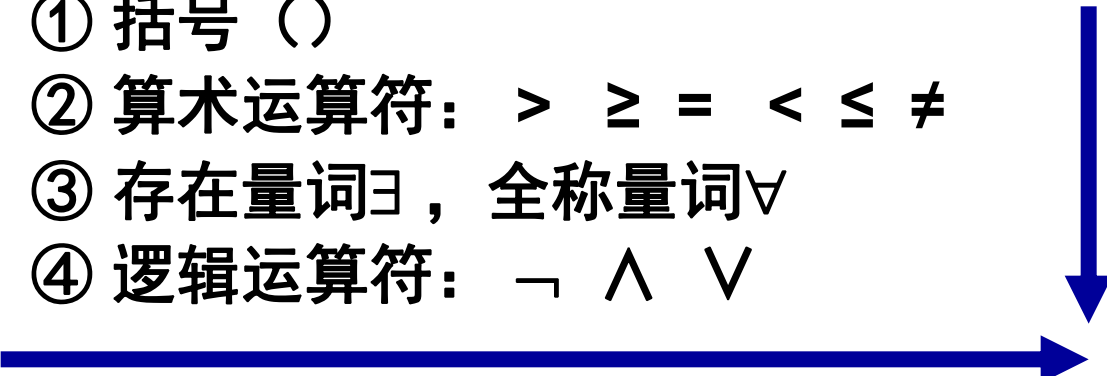
- 其中：
 - t —— 元组变量。表示一个元组，若 t 中有多个分量，表示为 $t[1]$, $t[2]$,;
 - $P(t)$ —— 由原子公式和运算符组成的复合公式。

2.5.1 元组关系演算

原子公式有下列三种形式：

- $R(s)$ R 是关系名， s 是元组变量。含义： s 是关系 R 的一个元组。
- $s[i] \theta u[j]$ s 和 u 是元组变量， θ 是算术比较运算符。含义：元组 s 的第 i 个分量与元组 u 的第 j 个分量之间满足 θ 关系。
- $s[i] \theta a$ 元组 s 的第 i 个分量值与常量 a 之间满足 θ 关系。

❖ 运算符包括四类：

- ① 括号 $()$
 - ② 算术运算符： $> \geq = < \leq \neq$
 - ③ 存在量词 \exists ，全称量词 \forall
 - ④ 逻辑运算符： $\neg \wedge \vee$
- 

2.5.1 元组关系演算

公式的递归定义如下：

- ① 原子公式 P 是一个公式。其值为 P 的真、假值。
- ② 如果 P_1 和 P_2 是公式，那么 $\neg P_1$ 、 $P_1 \wedge P_2$ 、 $P_1 \vee P_2$ 也是公式。其真假值遵循逻辑运算的一般原则。
- ③ 如果 P 是公式，那么 $(\exists t)P(t)$ 也是公式。设元组变量的域集 $T = \{t_1, t_2, \dots, t_n\}$ ， $(\exists t)P(t) \Leftrightarrow P(t_1) \vee P(t_2) \vee \dots \vee P(t_n)$ ，至少存在一个元组 t_i 使得公式 P 为真，否则为假。
- ④ 如果 P 是公式，那么 $(\forall t)P(t)$ 也是公式。设元组变量的域集 $T = \{t_1, t_2, \dots, t_n\}$ ， $(\forall t)P(t) \Leftrightarrow P(t_1) \wedge P(t_2) \wedge \dots \wedge P(t_n)$ ，所有元组 t_i 使得 P 为真时上式为真，否则为假。

t 在 P 中是自由变量，在 $(\exists t)P(t)$ 和 $(\forall t)P(t)$ 中是约束变量，即：
自由元组变量——在一个公式中 t 未用 \exists 、 \forall 符号定义；
约束元组变量——在一个公式中 t 用 \exists 、 \forall 符号定义；

2.5.1 元组关系演算

□ 例：设有两个关系 R 和 S，求表达式的值：

R

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

S

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

$$1) R_1 = \{ t \mid S(t) \wedge t[1] > 2 \}$$

R₁

A	B	C
3	4	6
5	6	9

$$2) R_2 = \{ t \mid R(t) \wedge \neg S(t) \}$$

R₂

A	B	C
4	5	6
7	8	9

$$3) R_3 = \{ t \mid (\exists u)(S(t) \wedge R(u) \wedge t[3] < u[1]) \}$$

R₃

A	B	C
1	2	3
3	4	6

2.5.1 元组关系演算

R

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

S

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

$$4) R_4 = \{ t \mid (\forall u)(R(t) \wedge S(u) \wedge t[3] > u[1]) \}$$

R₄

A	B	C
4	5	6
7	8	9

$$5) R_5 = \{ t \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge u[1] > v[2] \wedge t[1] = u[2] \wedge t[2] = v[3] \wedge t[3] = u[1]) \}$$

R₅

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

应用

S (Sno, Sname, Ssex, Sage, Sdept)

C (Cno, Cname, Cpno, Ccredit)

SC (sno, cno, grade)

例1：查询CS系的学生信息

$\{ t \mid S(t) \wedge t[5]='CS' \}$

例2：查询学习课程号为2的学生学号。

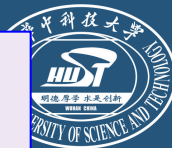
$\{ t \mid (\exists u)(SC(u) \wedge u[2]='2' \wedge t[1]=u[1]) \}$

例3：查询选修了“数据库”课程的学生学号。

$\{ t \mid (\exists u)(\exists v)(SC(u) \wedge C(v) \wedge v[1]=u[2] \wedge v[2]='数据库' \wedge t[1]=u[1]) \}$

例4：查询选修了全部课程的学生学号。

$\{ t \mid (\forall u)(C(u) \wedge (\exists v)(SC(v) \wedge v[2]=u[1] \wedge t[1]=v[1])) \}$



2.5.2 域关系演算*

- 域关系演算类似于元组关系演算，不同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。域演算表达式形为：

$$\{ t_1 \dots t_k \mid P(t_1, \dots, t_k) \}$$

其中 $P(t_1, \dots, t_k)$ 是关于自由域变量 t_1, \dots, t_k 的公式。

- 域关系演算的公式中也可使用 \wedge 、 \vee 、 \neg 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。

- 域演算的原子公式：

① $R(x_1 \dots x_k)$ ： R 是一个 k 元关系， x_i 是常量或域变量。含义：由 x_1, \dots, x_k 组成的元组在关系 R 中。

② $x \theta y$ ： 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。含义： x 和 y 之间满足关系 θ 。

例：设有R、S和W三个关系，求表达式的值：

R

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

S

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

W

D	E
7	5
4	8

1) $R_1 = \{xyz \mid R(xyz) \wedge x < 5 \wedge y > 3\}$

R₁

A	B	C
4	5	6

2) $R_2 = \{xyz \mid R(xyz) \vee (S(xyz) \wedge y = 4)\}$

R₂

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

3) $R_3 = \{xyz \mid (\exists u)(\exists v)(R(zxu) \wedge W(yv) \wedge u > v)\}$

R₃

A	B	C
5	7	4
8	7	7
8	4	7

应用示例

例1：查询计算机系(IS)的全体学生。

$\{ abcde \mid S(abcde) \wedge e = 'CS' \}$

S (Sno, Sname, Ssex, Sage, Sdept)

C (Cno, Cname, Cpno, Ccredit)

SC (sno, cno, grade)

例2：查询年龄小于20岁的学生。

$\{ abcde \mid S(abcde) \wedge d < 20 \}$

例3：检索选修课程号为5的学生学号和姓名。

$\{ ab \mid (\exists u)(\exists v)(S(abcde) \wedge SC(uvw) \wedge a=u \wedge v='5') \}$

关系运算的安全性(★)

- 关系演算有可能会产生无限关系和无穷验证，这样的表达式是不安全的。例如： $\{t \mid \neg R(t)\}$ ， $(\forall u)(\omega(u))$ 。
- 不产生无限关系和无穷验证的运算称为**安全运算**，其运算表达式称为**安全表达式**，所采取的措施称为**安全限制**。
- 在关系演算中，引入公式P的域概念，用**DOM(P)**表示。
DOM(P) = 显式出现在P中的值 + 在P中出现的关系的元组中出现的值（不必是最小集）
- 满足下列条件时，称元组演算表达式 $\{t \mid P(t)\}$ 是安全的：
 - 出现在表达式 $\{t \mid P(t)\}$ 结果中的所有值均来自DOM(P)；
 - 对P中的每个形如 $(\exists u)(\omega(u))$ 的子式，若u使 $\omega(u)$ 为真，则u的每个分量必属于DOM(P)。
 - 对P中的每个形如 $(\forall u)(\omega(u))$ 的子式，若u使 $\omega(u)$ 为假，则u的每个分量必属于DOM(P)。

关系运算的安全性(★)

R

A	B
a1	b1
a2	b2

S

A	B
1	d
5	b
6	c
7	d

$$R_1 = \{ t \mid \neg R(t) \}$$

$$\text{DOM}(P) = \{ \{a1, a2\}, \{b1, b2\} \}$$

R₁

A	B
a2	b1
a1	b2

$$R_2 = \{ t \mid (\exists u)(S(u) \wedge u[1] > 3 \wedge t[1] = u[2]) \}$$

$$\text{DOM}(P) = \{ \{1, 5, 6, 7, 3\}, \{d, b, c\} \}$$

R₂

B
b
c
d

元组演算对基本关系操作的表示

并: $R \cup S \equiv \{ t \mid R(t) \vee S(t) \}$

差: $R - S \equiv \{ t \mid R(t) \wedge \neg S(t) \}$

笛卡儿积: $R \times S \equiv \{ t(r+s) \mid (\exists u)(\exists v)(R(u) \wedge S(v) \wedge$

$$t[1]=u[1] \wedge \dots \wedge t[r]=u[r] \wedge$$

$$t[r+1]=v[1] \wedge \dots \wedge t[r+s]=v[s])$$

投影: $\pi_{i_1, \dots, i_m}(R) \equiv \{ t(m) \mid (\exists u) R(u) \wedge t[1]=u[i_1] \wedge$

$$t[2]=u[i_2] \wedge \dots \wedge t[m]=u[i_m] \}$$

选择: $\sigma_F(R) \equiv \{ t \mid R(t) \wedge F' \}$ (F' 是由 F 变化形成的谓词公式)

$(\exists t)P(t)$
 $(\forall t)P(t)$

域演算对基本关系操作的表示

并: $R \cup S \equiv \{ x_1 x_2 \dots x_n \mid R(x_1 x_2 \dots x_n) \vee S(x_1 x_2 \dots x_n) \}$

差: $R - S \equiv \{ x_1 x_2 \dots x_n \mid R(x_1 x_2 \dots x_n) \wedge \neg S(x_1 x_2 \dots x_n) \}$

笛卡儿积: $R \times S \equiv \{ x_1 x_2 \dots x_n y_1 y_2 \dots y_m \mid$
 $R(x_1 x_2 \dots x_n) \wedge S(y_1 y_2 \dots y_m) \}$

投影: $\pi_{i_1, \dots, i_m}(R) \equiv \{ x_{i_1} x_{i_2} \dots x_{i_m} \mid R(x_1 x_2 \dots x_n) \}$

选择: $\sigma_F(R) \equiv \{ x_1 x_2 \dots x_n \mid R(x_1 x_2 \dots x_n) \wedge F' \}$
(F' 是由 F 变化而形成的谓词公式)

$(\exists u)(\exists v)(R(zxu) \wedge W(yv) \wedge u > v)$

2.6 小结

本章作业

P70 5, 6 (仅用关系代数完成)

发布一周内完成

□ 关系数据库系统与非关系数据库系统的区别:

- 关系系统只有“表”这一种数据结构;
- 非关系数据库系统还有其他数据结构, 以及对这些数据结构的操作

□ 关系模型概述

- 关系数据结构及定义 (关系, 候选码, 主码, 外码, 关系模式, 关系数据库、关系数据库模式 (型-值))。
- 关系操作 (查询、插入、删除、修改: 集合操作)
- 关系的完整性约束 (实体完整性, 参照完整性, 用户定义的完整性)。
- **关系代数** (5个基本运算 (并, 差, 笛卡儿积, 选择, 投影) 以及交, 连接, 自然连接, 除)
- 关系演算* (元组关系演算, 域关系演算, 关系运算安全性DOM)