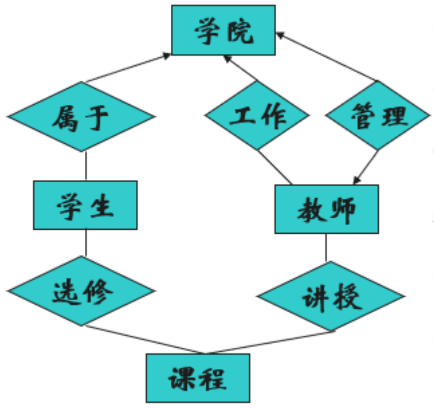
通用样例



学院D(dno，dname，dean)

学生S(sno，sname，gender，age，dno)

课程C(cno，cname，credit)

教师T(tno，tname, dno，sal)

学生选课SC(sno，cno，score)

老师教课TC(tno，cno)

关系模型

关系的基本概念

域（Domain）

- 一组值的集合，这组值**具有相同的数据类型**
- 域的基数：集合的值的个数

笛卡尔积(Cartesian Product)

- 一组域$D_1, D_2, \ldots, D_n$的笛卡尔积为:

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

- 笛卡尔积的每个元素$(d_1, d_2, \ldots, d_n)$称作一个n-元组(n-tuple)
- 元组的每一个值 d_i 叫做一个分量(component)
- 若D_i的基数为m_i，则笛卡尔积的基数为$\prod_{i=1}^{n} m_i$

– 例：

D_1为教师集合(T) = {t_1，t_2}

D_2为学生集合(S) = {s_1，s_2，s_3}

D_3为课程集合(C) = {c_1，c_2}

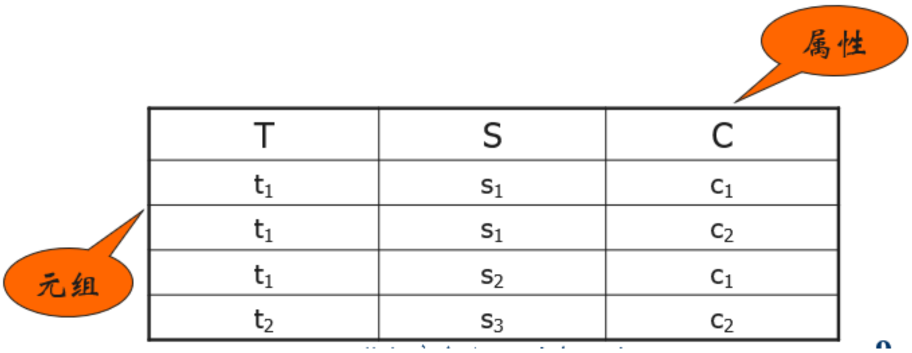
则$D_1 \times D_2 \times D_3$是个三元组集合，元组个数（基数）为$2 \times 3 \times 2$，是所有可能的(教师，学生，课程)元组集合

– 笛卡尔积可表示为二维表的形式

T	S	C
t_1	s_1	c_1
t_1	s_1	c_2
t_1	s_2	c_1
…	…	…
t_2	s_3	c_2

关系

- **笛卡尔积$D_1 \times D_2 \times \ldots \times D_n$的子集叫做在域$D_1, D_2, \ldots, D_n$上的关系**，用$R(D_1, D_2, \ldots, D_n)$表示
- R是关系的名字，n是关系的度、目或者元
- 关系是笛卡尔积中**有意义的子集**
- 关系也可以表示为二维表



T	S	C
t_1	s_1	c_1
t_1	s_1	c_2
t_1	s_2	c_1
t_2	s_3	c_2

注意

- 当关系作为关系数据库的数据结构时，需要对其进行限定，使其满足：
  - 无限关系在数据库中是无意义的
  - 通过为关系的每个列附加一个属性名的方法取消关系属性的有序性，即

$$(d_1, d_2, \ldots, d_i, d_j, \ldots, d_n) = (d_1, d_2, \ldots, d_j, d_i, \ldots, d_n)$$

- 本课程中提到的关系是被限定的“关系”。

关系的性质

- 列是同质的
  - 即**每一列中的分量来自同一域，是同一类型的数据。**
    - 如$TEACH(T, S, C) = (t_1, s_1, c_1), (t_1, t_2, c_1)$是错误的
- **不同的列可来自同一域，每列必须有不同的属性名。**
  - 如$P = \{t_1, t_2, s_1, s_2, s_3\}$，$C = \{c_1, c_2\}$，则$TEACH$不能写成$TEACH(P, P, C)$，还应写成$TEACH(T, S, C)$
- **列的次序可以任意交换**
  - 遵循这一性质的数据库产品(如ORACLE)，增加新属性时，永远是插至最后一列
  - 但也有数据库产品没有遵循这一性质，例如FoxPro仍然区分了属性顺序
- **任意两个元组不能完全相同**
  - 由笛卡尔积的性质决定
  - 但许多关系数据库产品没有遵循这一性质。例如：Oracle，DB2等都允许关系表中存在两个完全相同的元组，除非用户特别定义了相应的约束条件
- **每一分量必须是不可再分的数据。**满足这一条件的关系称作满足第一范式(1NF)的

关系模式

关系的描述称作关系模式，包括关系名、关系中的属性名、属性向域的映象、属性间的数据依赖关系等

- 关系模式可以形式化地表示为：$R(U,D,dom,F)$
  - R　关系名
  - U　组成该关系的属性名集合
  - D　属性组U中属性所来自的域
  - dom 属性向域的映象集合
  - F　属性间的数据依赖关系集合
- **关系是某一时刻的值，是随时间不断变化的**
- **关系模式是型，是稳定的**

三类关系

- **基本关系(基本表或基表)：实际存在的表，是实际存储数据的逻辑表示**
- 查询表：查询结果对应的表
- 视图
  - 视图：由基本表或其他视图表导出的表，是虚表，不对应实际存储的数据
  - 物化视图：由基本表或其他视图表导出的表，存储数据

关系数据库

- 其型是关系模式的集合，即数据库描述，称作数据库的内涵(Intension)
- 其值是某一时刻关系的集合，称作数据库的外延(Extension)

码

分类

超码(superkey)

- 是一个或多个属性的集合，这些属性的集合**可以在一个关系中唯一地标识一个元组**
- 一个关系可以有多个超码
  - 以$s(sno,sname,age,id)$为例，可能的超码包括$(sno),(id),(sno,sname),(sno,sname,id),(sno,sname,age,id)$等

候选码(Candidate Key)

- 是一个或多个属性的集合，**其值能唯一标识一个元组，其任意真子集均不是超码**，这样的属性集合称作候选码。
- 一个关系可以有多个候选码
- 候选码是最小的超码
  - 以$s(sno,sname,age,id)$为例，候选码只能是(sno)和(id)

主码(Primary Key)

- 进行数据库设计时，**从一个关系的多个候选码中选定一个作为主码**，如可选定sno作为关系S的主码

外部码(Foreign Key)

- **关系R中的一个属性组，它不是R的主码，但它与关系S的主码相对应**，则称这个属性组为R的外部码(R和S可以是同一关系)。

如：

学生(学号，姓名，性别，院系编号，年龄)
院系(院系编号，专业名)

如何确定超码

- 按照现实世界语义约束定义
- 不能依据对数据的归纳总结定义
- 应该选择其值从不变化或者很少变化的属性

数据库完整性

数据库完整性(`Database Integrity`)是指**数据库中数据的正确性和相容性。**

数据库完整性由各种各样的完整性约束来保证，是数据库设计的重要组成部分

数据库完整性约束可以通过DBMS或应用程序来实现

- 基于DBMS的完整性约束作为模式的一部分存入数据库中
- 由应用软件实现的数据库完整性则纳入应用软件设计

数据库完整性主要作用：

- 防止合法用户使用数据库时向数据库中添加不合语义的数据 。
- 利用基于DBMS的完整性控制机制来实现业务规则，易于定义，容易理解，而且可以降低应用程序的复杂性，提高应用程序的运行效率。
- 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。

关系模式的完整性

- 实体完整性(Entity Integrity)
  - 关系的**主码中的属性值不能为空值**
    - 意义：关系对应到现实世界中的实体集，元组对应到实体，实体是相互可区分的，通过主码来唯一标识，若主码为空，则**出现不可标识的实体，这是不允许的**
    - 实体完整性规则规定基本关系的主码中的所有属性都不能取空值
- 参照完整性(引用完整性，Referential Integrity)
  - 在关系模型中实体及实体间的联系都是用关系来描述的，因此可能存在着关系与关系间的引用。
  - **如果关系R的外部码F_k与关系S的主码P_k相对应(R和S可以是同一个关系)，则R中的每一个元组的F_k值或者等于S中某个元组的P_k值，或者为空值.**
    - 也就是说，**如果关系R的某个元组t2参照了关系S的某个元组t1，则t1必须存在**
- 空值：不知道或无意义
- 注意：

    - **数据库中所有的数据类型取值均可为null**
    - 空值不是0或者空字符串
- 空值的表现
    - 参与算术运算：结果为null
    - **参与比较运算：结果为null**
    - 参与逻辑运算：
        - 1、null or true=true
        - 2、null and false=false
        - 3、其它情况结果为null

变量a取值	运算公式	运算结果
10	a=null	null
10	a<>null（a不等于null）	null
10	a is null	false
10	a is not null	true

关系数据语言(关系代数及其拓展都特别重要)

是抽象的语言。

基本运算 - 选择(select)

定义

在关系R中选择满足给定条件的元组(从**行**的角度)：如 $\sigma_F(R) = \{t|t \in R \wedge F(t) = true\}$

F是选择的条件，$\forall t \in R$，$F(t)$要么为真，要么为假

F的形式：由逻辑运算符连接关系表达式而成

- 逻辑表达式： 如 $\wedge, \vee, \neg$
- 关系表达式：$X\theta Y$
    - X，Y是属性名、常量、或算术表达式
    - θ 是比较运算符，$\theta \in \{>, \geq, <, \leq, =, <>\}$

基本运算 - 投影(project)

定义

从关系R中取若干列组成新的关系(从**列**的角度)

$\Pi_{A_i}(R) = \{t[A_i]|t \in R\}, A_i \subseteq U$ ，(**U是关系R中的所有属性的集合**)

投影的结果中要去掉相同的元组

最后留下的结果集就是A_i

样例

如：查询001号学生所选修的课程号

$\Pi_{cno}(\sigma_{sno='001'}(SC))$

这被称之为复合关系代数运算。也就是说，运算只要作用在关系上即可。

基本运算 - 笛卡尔积运算

元组的连串(Concatenation)

若$r=(r_1,\ldots,r_n), s=(s_1,\ldots,s_m)$，则定义r与s的连串为：

$\widehat{rs}=(r_1,\ldots r_n,s_1,\ldots,s_m)$

定义

两个关系 R，S，其度分别为 n，m，则它们的笛卡尔积(**广义**)是所有这样的元组集合：元组的前 n 个分量(属性)是 R 中的一个元组，后 m 个分量(属性)是 S 中的一个元组

$R\times S=\{\widehat{rs}|r\in R\wedge s\in S\}$

$R\times S$**的度为R与S的度之和，**$R\times S$**的元组个数为** R **和** S **的元组个数的乘积。**

注意：关系的度和元是同样的概念。

举例

R

A	B
α	1
β	2

S

A	D	E
α	10	a
β	10	a
β	20	b
γ	10	b

R x S

R.A	B	S.A	D	E
α	1	α	10	a
α	1	β	19	a
α	1	β	20	b
α	1	γ	10	b
β	2	α	10	a
β	2	β	10	a
β	2	β	20	b
β	2	γ	10	b

注意

- **运算结果的命名：关系名.属性名**
    - 不重名则可以忽略前缀，重名则一定不能忽略。
- 效率问题：

$$\prod_{s.sno,sname,cno,score}(\sigma_{s.sno = sc.sno \wedge s.dept = '软'}(S \times SC))$$

$$\prod_{s.sno,sname,cno,score}(\sigma_{s.sno = sc.sno}((\sigma_{s.dept = '软'}(S)) \times SC))$$

下面的查询语句好，因为上面的语句会生成特别庞大的笛卡尔积结果作为中间值。

附加运算 - θ 连接(θ - join)

定义

$$R \underset{A\theta B}{\bowtie} S = \{\widehat{rs} | r \in R \wedge s \in S \wedge r[A]\theta r[S]\}$$

- A , B 为 R 和 S 上度数相等且可比的属性集合（A、B是两个域）
- θ 为比较运算符，θ 为等号时称为**等值连接**

$$R \underset{A\theta B}{\bowtie} S = \sigma_{r_{[A]\theta[B]}}(R \times S)$$

附加运算 - 自然连接(natural-join)

定义

- 从两个关系的笛卡儿积中选取在**相同属性列B上取值相等的元组**，**并去掉重复的属性。**
- **如果是多个相同属性列，则对应元组的多个属性列都应该取值相等**

$$R \bowtie S = \{\widehat{rs} | r \in R \wedge s \in S \wedge r[B] = s[B]\}$$

- 自然连接与等值连接的不同
  - 自然连接中相等的分量必须是相同的属性集合，并且要在结果中去掉重复的属性(**两个关系中相同的属性在自然连接的结果关系模式中只出现一次**)，而等值连接则不必。
- 可交换，可结合
  - $S \bowtie SC \equiv SC \bowtie S$
  - $(S \bowtie SC) \bowtie C \equiv S \bowtie (SC \bowtie C)$
- **要注意参与自然连接的关系中是否有不希望做选择条件的同名属性**

如:

计算R $\bowtie$ S的结果

R

A	B	C	D
α	1	α	a
β	2	γ	a
γ	4	β	b
α	1	γ	a
δ	2	β	b

S

B	D	E
1	a	α
3	a	β
1	a	γ
2	b	δ
3	b	δ

R $\bowtie$ S

A	B	C	D	E
α	1	α	a	α
α	1	α	a	γ
α	1	γ	a	α
α	1	γ	a	γ
δ	2	β	b	δ

基本运算 - 并运算(union)

定义

所有至少出现在两个关系中之一的元组集合：$R \cup S = \{t|t \in R \vee t \in S\}$

- **两个关系R和S若进行并运算，则它们必须是相容的**：
  - 关系R和S必须是同元的，即它们的属性数目必须相同
  - 对 $\forall i, R$ 的第 i 个属性的域必须和 S 的第 i 个属性的域相同
  - 保证同质：同列数据的数据类型相同
- 并运算会去重

基本运算 - 差运算(difference)

定义

所有出现在一个关系而不在另一关系中的元组集合：$R - S = \{t|t \in R \wedge t \notin S\}$

- **R和S必须是相容的**

样例

查询选修了c1号而没有选c2号课程的学生学号

$$\Pi_{sno}(\sigma_{cno = 'c1'}(SC)) - \Pi_{sno}(\sigma_{cno = 'c2'}(SC))$$

查询未选修c1号课程的学生学号

方案1：$\Pi_{sno}(S) - \Pi_{sno}(\sigma_{cno = 'c1'}(SC))$?

方案2：$\Pi_{sno}(\sigma_{cno <> 'c1'}(SC))$?

方案3：$\Pi_{sno}(SC) - \Pi_{sno}(\sigma_{cno = 'c1'}(SC))$?

方案一正确。方案二错误，一个学生可能既选c1也选c2。方案三不全，可能有没选课的学生。

附加运算 - 交运算(intersection)

定义

所有同时出现在两个关系中的元组集合：$R \cap S = \{t|t \in R \wedge t \in S\}$

- 交运算可以通过差运算来重写

$R \cap S = R - (R - S)$

- **参与交运算的关系必须相容**

基本运算的分配律

- 投影和并可以分配

$$\Pi_{pid,name}(S \cup T) \equiv \Pi_{pid,name}(S) \cup \Pi_{pid,name}(T)$$

- 投影和差不可分配

$$\Pi_{pid,name}(S - T) \neq \Pi_{pid,name}(S) - \Pi_{pid,name}(T)$$

不可分配的原因：对于重复元素，左面会留一个，右面一个不留。

附加运算 - 赋值运算(assignment)

定义

- 为使查询表达简单、清晰，可以将一个复杂的关系代数表达式分成几个部分，每一部分都赋予一个临时关系变量，该变量可被看作关系而在后面的表达式中使用

$\text{临时关系变量} \leftarrow \text{关系代数表达式}$

- **赋值给临时关系变量只是一种结果的传递，而赋值给永久关系则意味着对数据库的修改**

基本运算 - 更名运算(rename)

定义

- 背景：关系代数表达式的运算结果没有可供我们引用的名字
- 更名运算：
  - $\rho_x(E)$，返回关系代数表达式 E 的运算结果，并把名字 x 赋给 E 的运算结果
  - $\rho_{x(A_1,A_2,...,A_n)}(E)$，返回表达式 E 的运算结果，并把名字 x 赋给E的运算结果，同时将各属性更名为$A_1, A_2, \ldots, A_n$
- 关系被看作一个最小的关系代数表达式，可以将更名运算施加到关系上，得到具有不同名字的同一关系。这样一个关系在同一个表达式中出现多次时是必须的
- 对于更名$\rho_{R_1}(R)$，更名完成后 R 仍然存在，类似于$R_1 = copy(R)$

附加运算 - 除运算(division)

除运算解决的问题：关系之间的包含(即 $R \subseteq S$)

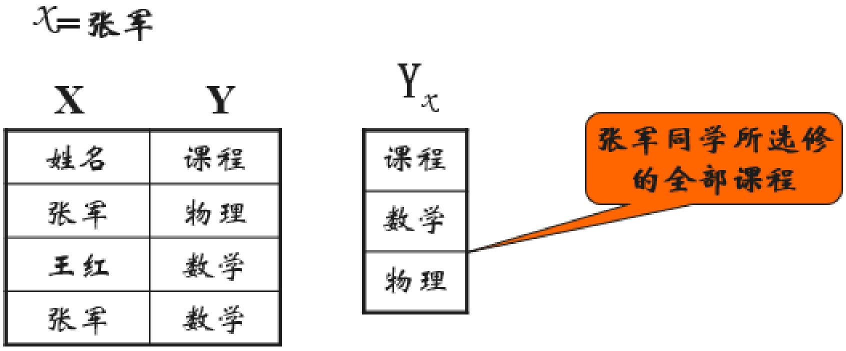
象集(Image Set)

定义

现有关系$R(X, Y)$，其中 X, Y 是属性集合，χ 是 X 上的取值，定义 χ 在 R 中的象集为：
$Y_\chi = \{t[Y] | t \in R \wedge t[X] = \chi\}$

从 R 中选出在 X 上取值为 χ 的元组，去掉 X 上的分量，只留 Y 上的分量

样例



又比如

R

A	B	C
a1	b1	c2
a2	b3	c7
a3	b4	c6
a1	b1	c3
a4	b6	c6
a2	b2	c3
a1	b2	c1

$X = \{A, B\}, \chi = \{a1, b1\}$，那么$R_\chi = \{c2, c3\}$。（$c2, c3$也可以分别加上括号）。

除运算的定义

给定关系$R(X，Y)$ 和$S(Y)$，其中X, Y为属性集合。

R **中的** Y **与** S **中的** Y **可以有不同的属性名，但必须出自相同的域。** 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 X 属性集合上的**投影**（Π）：

元组在 X 上分量值 χ 的象集 Y_χ **包含** S **在** Y **上投影的集合。**

$$R \div S = \{t[X] | t \in R \wedge \Pi_Y(S) \subseteq Y_\chi\}$$

Y_χ：χ在 R 中的象集，$\chi = t[X]$

换言之，对于关系 $R(X, Y), S(Y)$，关系$R \div S$ 是属性集合 X上的关系，元组 t 属于 $R \div S$，当且仅当以下条件成立：

- $t \in \Pi_X(R)$
- 对于 S 中的**任意一个**元组t_s，在 R 中都有元组 t_R 同时满足以下条件：
  - $t_R[Y] = t_s[Y]$
  - $t_R[X] = t$

举例

R

A	B	C
a1	b1	c2
a2	b3	c7
a3	b4	c6
a1	b1	c3
a4	b6	c6
a2	b2	c3
a1	b2	c1

a1的象集 {(b1,c2),(b2,c3),(b2,c1)}

a2的象集 {(b3,c7),(b2,c3)}

a3的象集 {(b4,c6)}

a4的象集 {(b6,c6)}

S

B	C
b1	c2
b2	c1
b2	c3

R÷S = {a1}

如何查询选修了全部课程的学生学号？

思路：逐个考虑选课关系SC中的元组t，计算t在学号sno上的分量x，再查询x在选课关系中的象集课程C_x，若C_x包含了所有的课程C，则x是满足条件的一个元组

选修全部课程的学生

X同学所选修的全部课程

全部课程

$$\{ x \mid x=t[sno] \wedge t \in SC \wedge C_x \supseteq C \}$$

请用关系代数表达式描述：查询选修了全部课程的学生的学号(使用÷)

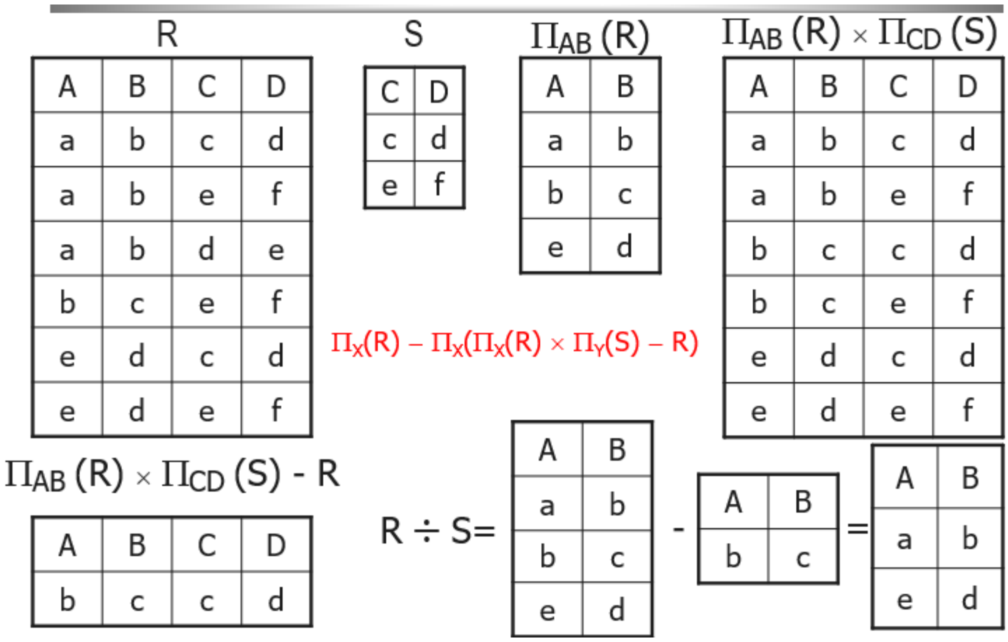
$$\Pi_{sno,cno}(SC) \div \Pi_{cno}(C)$$

用其他关系表达式表示除运算

$R(X,Y) \div S(Y) = \Pi_X(R) - \Pi_X(\Pi_X(R) \times \Pi_Y(S) - R)$

也就是说，首先从关系 R 中投影选出仅包含 `x` 的关系，关系中的某些元组可能并没有完全包含$S(Y)$，即 $\Pi_Y(S) \subseteq Y_X$ 不成立，所以要将这一部分剔除。我们构造 R 在 X 上的投影与 S 在 Y 上的投影这两部分的笛卡尔积，然后减去关系 R ，就是不满足上述条件的部分。

如：



R

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

S

C	D
c	d
e	f

$\Pi_{AB}(R)$

A	B
a	b
b	c
e	d

$\Pi_{AB}(R) \times \Pi_{CD}(S)$

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

$\Pi_X(R) - \Pi_X(\Pi_X(R) \times \Pi_Y(S) - R)$

$\Pi_{AB}(R) \times \Pi_{CD}(S) - R$

A	B	C	D
b	c	c	d

$R \div S =$

A	B
a	b
b	c
e	d

\-

A	B
b	c

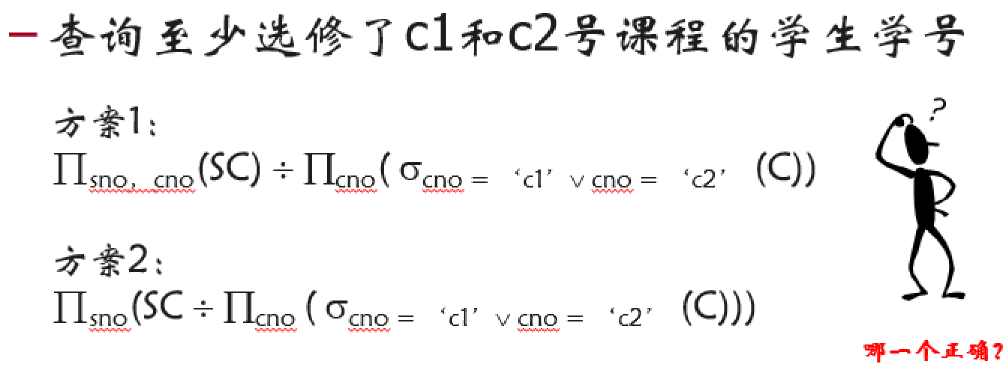
=

A	B
a	b
e	d

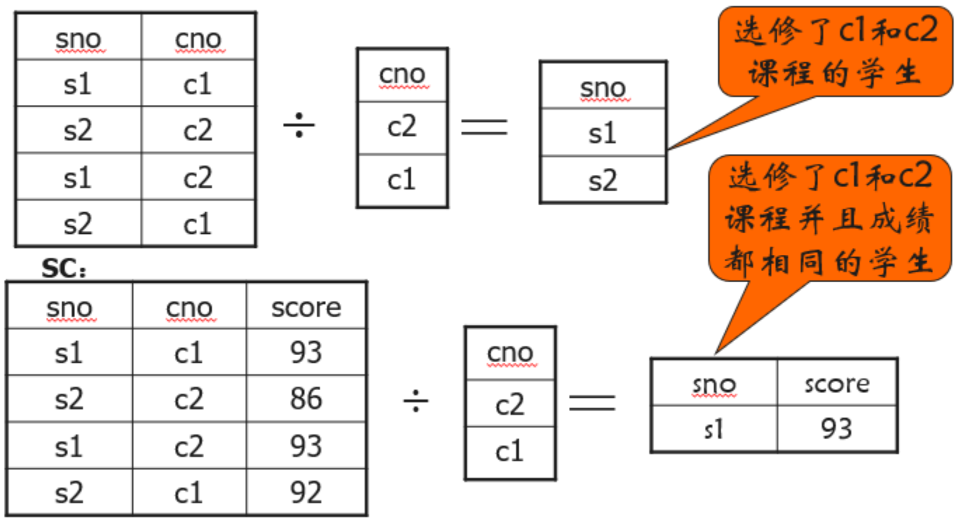
注意的问题

提前使用投影运算删除影响除运算结果的属性

如:



方案一是正确的，方案二不正确因为存在"分数"这个属性没有被排除导致在做除法时属性集合`x`包含了不应包含的属性。



sno	cno
s1	c1
s2	c2
s1	c2
s2	c1

÷

cno
c2
c1

=

sno
s1
s2

SC:

sno	cno	score
s1	c1	93
s2	c2	86
s1	c2	93
s2	c1	92

÷

cno
c2
c1

=

sno	score
s1	93

关系代数对于空值的处理

以该关系为例:

S

sno	sname	dno	age
S1	甲	计	20
S2	乙	软	21
S3	丙	软	
S4	丁	软	

- $\sigma_F(E)$
  - **保留使 F 确定的为真的元组**
    - $\sigma_{age=20}(S)$或者$\sigma_{age<>20}(S)$
- $\Pi_{A_1,A_2,\cdots,A_n}(E)$
  - **元组表现相同(认为表示的语义相同)，则保留一个元组**
    - 查询各系年龄分布
    - $\Pi_{dno,age}(S)$，会保留三个元组
- $\cap \cup -$ 运算与 Π 的处理原则一致

扩展的关系代数

外连接（Outer Join）

问题引入

– 例：查询老师的有关信息，包括教师编号、姓名、工资、所教授的课程编号和名称

$$\Pi_{tno,tname,sal,cno,cname}(T \bowtie TC \bowtie C)$$

T

TNO	TNAME	SAL
P01	赵明	800
P02	钱广	700
P03	孙立	600
P04	李三	500

$\bowtie$

TC

CNO	TNO
C01	P01
C02	P02
C02	P04

$\bowtie$

C

CNO	CNAME
C01	物理
C02	数学
C03	化学

=

TNO	TNAME	SAL	CNO	CNAME
P01	赵明	800	C01	物理
P02	钱广	700	C02	数学
P04	李三	500	C02	数学

问题：有关P03号职工的姓名和工资信息没有显示出来

TC 中没有 $P03$ 导致在做自然连接时 $P03$ 被舍弃，如何能在结果中显示 $P03$？

定义

为避免自然连接时因失配而发生的信息丢失，可以**假定在参与连接的一方关系中附加一个取值全为空值的元组，它和参与连接的另一方关系中的任何一个未匹配上的元组都能匹配，称之为外连接**

外连接 = 自然连接 + 失配的元组

外连接的形式

左外连接、右外连接、全外连接

- $⟕$ 左外连接 = 自然连接 + 左侧关系中失配的元组
- $⟖$ 右外连接 = 自然连接 + 右侧关系中失配的元组
- $⟗$ 全外连接 = 自然连接 + 两侧关系中失配的元组

[数据库常用关系代数符号在 LaTeX 中的表示]

广义投影(Generalized Projection)

定义

在投影列表中使用**算术表达式**来对投影进行扩展$\Pi_{F_1,F_2,...,F_n}(E)$。

其中：

- E 是关系代数表达式
- $F_1, F_2, \cdots, F_n$ **是涉及常量以及 E 中属性的算术表达式**

示例

$\rho_{TAX(tno,income-tax)}(\Pi_{tno,sal*5/100}(T))$

聚集函数(Aggregate Functions)

定义

- 计算给定关系的统计信息，**返回单一值**
- 使用聚集函数的关系**可以是多重集(multiset)，即一个元组可以出现多次。**
- 如果想去除重复元组，可以用连接重复符 '-' 将 'distinct' 附加在聚集函数名后，如sum-distinct是去重的求和

函数

- `sum`：求和
  - 计算001号学生的总成绩
    - $\mathcal{G}_{sum(score)}(\sigma_{sno='001'}(SC))$
- `avg`：计算平均值
  - 查询001号同学选修课程的平均成绩。
    - $\mathcal{G}_{avg(score)}(\sigma_{sno='001'}(SC))$
- `max`：计算最大值， `min`：计算最小值
  - 查询学生选修数学课程的最高成绩
    - $\mathcal{G}_{max(score)}(\sigma_{cname='数学'}(C) \bowtie SC))$

- `count`：计数（要分清计数与求和）
  - 查询学生总人数
    - $\mathcal{G}_{count(sno)}(S)$或$\mathcal{G}_{count(*)}(S)$ （*：通配，每个元组加一）
  - 查询选课学生的总人数
    - $\mathcal{G}_{count-distinct(sno)}(SC)$
  - 查询选课学生的总人次
    - $\mathcal{G}_{count(sno)}(SC)$

聚集函数的分组

将一个关系中的元组分为若干个组，在每个分组上使用聚集函数。

$_{\text{属性下标}}\mathcal{G}_{\text{聚集函数}(\text{属性下标})}(\text{关系})$

- 第一个属性下标：按此属性上的值对关系分组。
- 第二个属性下标：对此属性在每个分组上运用聚集函数。

聚集函数分组运算的一般形式

$_{G_1,G_2,\ldots,G_n}\mathcal{G}_{F_1(A_1),F_2(A_2),\ldots,F_m(A_m)}(E)$

G_i 是用于分组的属性，F_i 是聚集函数，A_i 是属性名。

G_i 将 E 的运算结果分为若干组，满足：

- 同一组中所有元组在 G_1 ,G_2 ，... , G_n 上的值相同。
- 不同组中元组在 G_1 ,G_2 ，... , G_n 上的值不同

最终结果集，每个属性下标都会成为结果集（关系）的一个属性

聚集函数对空值的处理

- 多重集中忽略null
- 聚集函数作用于空集合：
  - $count(\Phi) = 0;$
  - 其它聚集函数作用于空集合，结果为null

样例

SC

Sno	Cno	Score
S1	C1	80
S1	C2	
S1	C3	80
S1	C4	95
S2	C1	
S2	C3	

Sno	Count(*)	Count(score)	Count-distinct(score)	Max(score)	Avg(score)
S1	4	3	2	95	85
S2	2	0	0		

注意的问题：

- 忽略null：所以 $S1$ 的分数平均分为85分。（$(80+80+95)\div 3$）
- 除了计数，聚集函数作用于空集合都是null

外连接进一步探索

- 查询每个学院女生的人数及学院名称(没有女生的学院也希望展现)
  - 方案1：$_{dno,dname}\mathcal{G}_{count(sno)}(D⟖(\sigma_{gender='F'}(S)))$
  - 方案2：$_{dno,dname}\mathcal{G}_{count(sno)}(\sigma_{gender='F'}(D⟖S))$

方案二是不正确的，因为在左外连接中，没有学生的学院被加进去了，但在进行选择时会被舍弃，导致没有女生的学院没有被展现。