



# 数据库原理及应用

主讲：陈安龙

*[chenanlong@uestc.edu.cn](mailto:chenanlong@uestc.edu.cn)*

电子科技大学

# 本章要求：

- 1、掌握关系、关系模式、关系数据库等基本概念
- 2、掌握关系的三类完整性的含义
- 3、掌握关系代数运算
- 4、PostgreSQL项目实践
- 5、挑战性问题 需要同学自己课外查阅资料自学，  
回答相关问题。

# 挑战性问题——数据库的数据模型

## 一、数据模型问题探讨

- 1、数据库系统与文件系统有哪些区别？
- 2、在数据模型中，如何表示数据之间的关系？
- 3、关系模型如何用数学方法定义与运算？
- 4、如何理解关系的完整性？它解决哪些问题？
- 5、关系模型优点与局限有哪些？

# 1、关系模型的基本概念

层次、网状数据库结构复杂，需要使用者有较强专业知识，使用很不方便。程序员必须经过良好的培训，对所使用的系统有深入的了解才能用好系统。

关系数据库是应用集合的方法来处理数据的。它具有结构简单、理论基础坚实、数据独立性高以及提供非过程性语言等优点。

# 关系数据模型-现有主流DBMS支持的逻辑模型

- **关系的概念**：用于描述**数据本身、数据之间联系**，俗称“表”。

学号	姓名	性别	专业号	年龄	班长
801	张三	女	01	19	802
802	李四	男	01	20	
803	王五	男	01	20	802
804	赵六	女	02	20	805
805	钱七	男	02	19	

字段名或属性名

行, 元组或记录

列, 字段或属性

- **域 (Domain)**：是一组具有**相同数据类型**的值的集合，具有数据类型及长度、域名、取值范围。
- **关系模式**：由关系名、各个属性、属性的域以及属性的依赖关系构成。
- **关系实例**：由真实记录或元组构成的集合，称为关系实例，简称关系；每个元组的字段必须对应关系模式中的字段。

## 一、关系的数学定义

**笛卡尔积**（Cartesian Product）：设**D1、D2、...、Dn**是**n**个域, 则它们的笛卡尔积为

$$D1 \times D2 \times \dots \times Dn = \{(d1, d2, \dots, dn) \mid di \in Di, i=1,2,\dots,n\}$$

其中每一个元素称为一个**n元组(n-tuple)**, 简称**元组**;

元组中的每个值**di**称为一个**分量(component)**.

笛卡尔积可以写成一个二维表

例如：

设  $D1=\{\text{张三}, \text{李四}\}$ ,

$D2=\{\text{数学}, \text{语文}\}$ ,

$D3=\{\text{优}, \text{良}\}$

则  $D1 \times D2 \times D3$  可用二维表表示为：

张三	数学	优
张三	数学	良
张三	语文	优
张三	语文	良
李四	数学	优
李四	数学	良
李四	语文	优
李四	语文	良

## 关系 (Relation)

笛卡尔积  $D1 \times D2 \times \dots \times Dn$  的子集合，

记作  $R(D1, D2, \dots, Dn)$

关系名

$n$  为关系的目或度

## 数学意义上的关系

(1) 笛卡尔积不满足交换律:

即:  $(d1, d2, \dots, dn) \neq (d2, d1, \dots, dn)$

(2) 数学意义的关系可以是无限个元组的集合。

数学意义的关系不适合数据库的实际应用, 必须做如下限制:

(1) 数据库关系模型中的关系必须是**有限的元组集合**

(2) 数据库关系模型中的属性列表的顺序是可交换的, 允许任意顺序; 具有数据库中**关系满足交换律**。



## 4、说明

- ① 关系是一个二维表。
- ② 每行对应一个元组。
- ③ 每列可起一个名字，称为属性。属性的取值范围为一个域，元组中的一个属性值是一个分量。

## 5、关系的性质

- ① 列是同质的，即每列中的数据必须来自同一个域
- ② 每一列必须是不可再分的数据项（不允许表中套表，即满足第一范式）
- ③ 不能有相同的行
- ④ 行、列次序无关

## 二、关系模型

三部分：关系数据结构、关系操作集合、关系的完整性

### （一）数据结构

#### 1、单一的数据结构：关系（二维表）

不论是实体还是实体间的联系都用关系表示。

实体值 → 关系的元组，在关系数据库中通常称为记录

属性值 → 元组的分量，在关系数据库中通常称为字段

关键字（码）：唯一标识一个元组的属性组

关键字可以有多个，统称候选关键字。在使用时，通常选定一个作为主关键字。主关键字的诸属性称为主属性，其它为非主属性。

## 2、关系模式：**关系的描述。**

包括关系名、诸属性名、属性域约束、属性间的依赖。

关系的型

类型、长度等

一个元组为关系的一个值，也称为记录

**关系数据库模式：**对关系数据库的描述，包括域的定义及在域上定义的所有**关系模式**。

型

**关系数据库：**所有实体及实体间联系的关系的集合。是某时刻所有关系模式对应的关系的集合。

值



# 简单地讲：关系数据模型是表现为二维表的形式

如：学生的基本信息

学号	姓名	住址	性别	兴趣爱好
20060101	张江	04-201	男	排球
20060102	魏明	04-203	男	足球
20060103	王昆	05-102	女	羽毛球
20060104	程香	05-102	女	羽毛球
20060105	刘鹏	04-405	男	游泳
20060106	王德启	04-203	男	排球
20060107	武飞	04-205	男	篮球
20060108	刘用	04-102	男	篮球
20060109	程文	05-304	女	乒乓球

### 3、关系的三种类型

基本关系：客观存在的基本表

查询表：由基本表按一定条件检索得到的结果

视图（View）：从一个或多个基本关系上导出的关系。它不对应实际的存储数据，是一个虚关系，然而可永久存在。相当于关系模型的外模式。

由于二维表的存储策略非常简单，关于数据库的物理存储完全由**DBMS自动完成**。因此，在关系模型中不需要与内模式相应的概念。



关系简单吗？

查询操作

## (二) 关系操作

1、种类：选择、投影、连接、除、并、交、差

增加、删除、更新

维护操作

2、特点：

① 集合操作，一次操作

一次一集合（关系型）

② 可存取多个元组

一次一记录（非关系型）

③ 非过程化语言：用户只需告诉做什么（What）

不需告诉怎么做（How）

④ 数据定义、数据操纵、数据控制语言集成在一起

DDL

DML

DCL：权限控制、完整性控制等

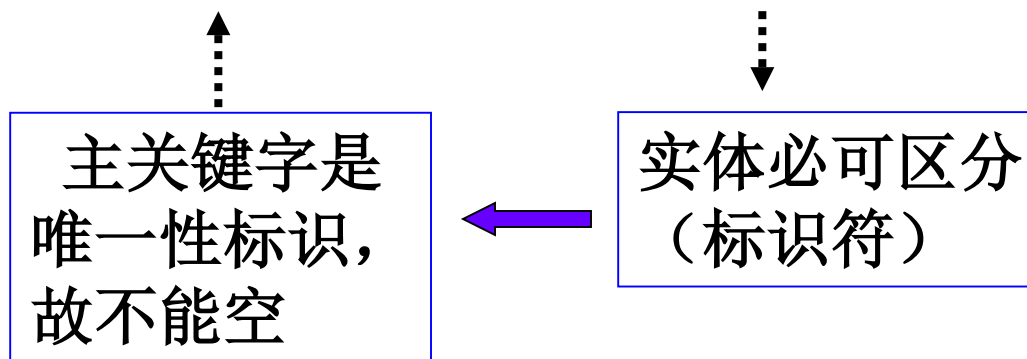


### (三) 关系模型的三类完整性

#### 1、实体完整性 (Entity Integrity)

基本关系的所有主属性不能取空值

原因：基本关系  $\longleftrightarrow$  实体集



#### 2、参照完整性 (Referential Integrity)，也叫引用完整性

若基本关系R含有与另一个基本关系S的主关键字相对应的属性组F (F称为R的**外键**或**外部码**)，则R中每个元组在F上的值或为空值，或等于S中某个元组的主关键字值。



例：职工关系 EMP (ENO, ENAME, **DNO**)

部门关系 DEPT (**DNO**, DNAME)

DEPT的主键

EMP的外键，  
只能取空值或  
DEPT中某关键字的值

又如：学生关系 (**SNO**, SNAME, AGE, SEX)

课程关系 (**CNO**, CNAME)

选课关系 (**SNO**, **CNO**, G)

### 3、用户定义的完整性

用户自定义完整性是针对某一具体关系数据库的约束条件，它反映某一具体应用所涉及的数据必须满足的语义要求。主要包括非空约束、唯一约束、检查约束、缺省值约束、外键约束

**注意：**定义完整性约束后，当数据库数据发生变化时，DBMS会自动检查，从而不必在应用程序中作检查

## 2 RDBS的数据操纵语言：关系代数

关系代数的运算对象是关系，运算结果也为关系。  
其运算按运算符的不同可分为两类。

### 一、传统的集合运算

1、并（Union）： $R \cup S = \{ t \mid t \in R \vee t \in S \}$

2、交（Intersection）： $R \cap S = \{ t \mid t \in R \wedge t \in S \}$

3、差（Difference）： $R - S = \{ t \mid t \in R \wedge t \notin S \}$

4、笛卡尔积（广义）： $R \times S = \{ \widehat{t_r t_s} \mid t_r \in R \wedge t_s \in S \}$

## (1) UNION(并): $R \cup S$

**概念：** 包含R和S中的所有元组，要求R和S兼容(字段个数、类型[名字])，结果模式与R一致。

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

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

$R \cup S$		
$A$	$B$	$C$
$a_1$	$b_1$	$c_1$
$a_1$	$b_2$	$c_2$
$a_2$	$b_2$	$c_1$
$a_1$	$b_3$	$c_2$

## (2) INTERSECT (交): $R \cap S$

概念：包含R、S中相同的元组，R、S须兼容。

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

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

$R \cap S$		
$A$	$B$	$C$
$a_1$	$b_2$	$c_2$
$a_2$	$b_2$	$c_1$

### (3) SET-DIFFERENCE(差): $R-S$

概念: 包含在 $R$ 中而不在 $S$ 中的元组,  $R$ 、 $S$ 兼容。

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

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

$R-S$		
$A$	$B$	$C$
$a_1$	$b_1$	$c_1$

## (4) CROSS-PRODUCT(积): $R \times S$

**概念：** 结果包含R和S中所有字段。如果有相同的字段名，则在结果字段来源的表。也叫“笛卡尔乘积”。

<i>R</i>		
<i>A</i>	<i>B</i>	<i>C</i>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>1</sub>	<i>c</i> <sub>1</sub>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>2</sub>
<i>a</i> <sub>2</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>1</sub>

<i>S</i>		
<i>A</i>	<i>B</i>	<i>C</i>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>2</sub>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>3</sub>	<i>c</i> <sub>2</sub>
<i>a</i> <sub>2</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>1</sub>

<i>R</i> × <i>S</i>					
<i>R.A</i>	<i>R.B</i>	<i>R.C</i>	<i>S.A</i>	<i>S.B</i>	<i>S.C</i>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>1</sub>	<i>c</i> <sub>1</sub>	<i>a</i> <sub>1</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>2</sub>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>1</sub>	<i>c</i> <sub>1</sub>	<i>a</i> <sub>1</sub>	<i>b</i> <sub>3</sub>	<i>c</i> <sub>2</sub>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>1</sub>	<i>c</i> <sub>1</sub>	<i>a</i> <sub>2</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>1</sub>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>2</sub>	<i>a</i> <sub>1</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>2</sub>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>2</sub>	<i>a</i> <sub>1</sub>	<i>b</i> <sub>3</sub>	<i>c</i> <sub>2</sub>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>2</sub>	<i>a</i> <sub>2</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>1</sub>
<i>a</i> <sub>2</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>1</sub>	<i>a</i> <sub>1</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>2</sub>
<i>a</i> <sub>2</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>1</sub>	<i>a</i> <sub>1</sub>	<i>b</i> <sub>3</sub>	<i>c</i> <sub>2</sub>
<i>a</i> <sub>2</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>1</sub>	<i>a</i> <sub>2</sub>	<i>b</i> <sub>2</sub>	<i>c</i> <sub>1</sub>

说明:

- ① 以上定义中,  $R$ 、 $S$ 本身也可以是关系代数表达式;
- ② 由于 $R \cap S = R - (R - S)$ , 故 $R \cap S$ 实际上是多余的。

# 专门的关系运算

学生-课程数据库： 学生Student、课程Course和选修SC

**Student**

学号 Sno	姓名 Sname	性别 Sex	年龄 Age	所在系 Dept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	IS
200215123	王敏	女	18	MA
200215125	张立	男	19	IS

**Course**

课程号 Cno	课程名 Cname	先行课 Cpno	学分 Ccredit
1	数据库	5	4
2	数学		2
3	信息系统	1	4
4	操作系统	6	3
5	数据结构	7	4
6	数据处理		2
7	PASCAL	6	4

**SC**

学号 Sno	课程号 Cno	成绩 Grade
200215121	1	92
200215121	2	85
200215121	3	88
200215122	2	90
200215122	3	80



# 选择与投影操作

## 1、选择（Selection），又称限制（Restriction）

从行的角度的运算

$\sigma_F(R)$ ：在关系R中选出满足条件F的元组形成新的关系。

条件表达式

## 2、投影（Projection）

从列的角度的运算

$\pi_A(R)$ ：在R中选出若干属性列组成一个新关系。

属性组

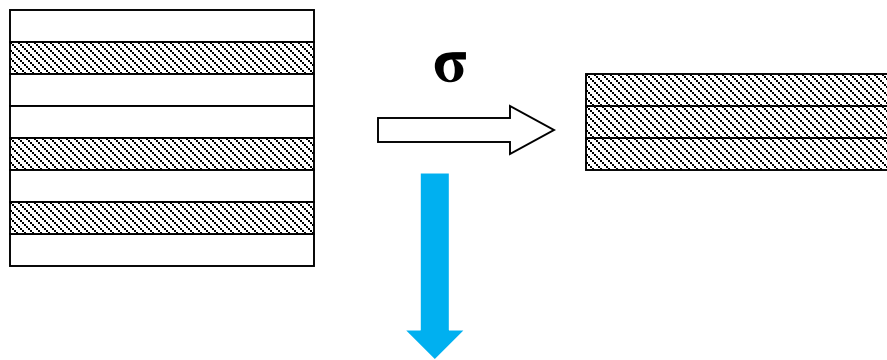
投影后若有重复行，则自动保留一个

共同点：为一元关系操作符。

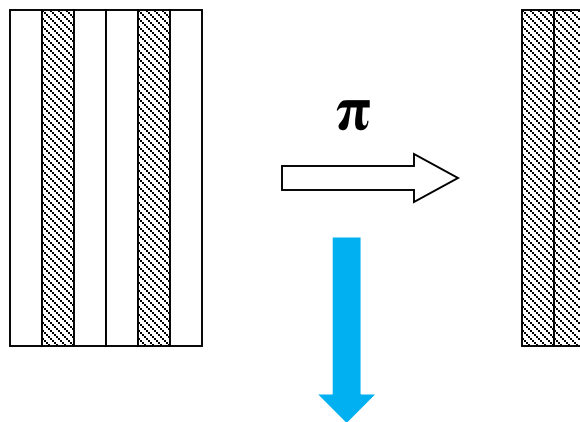
选择：从关系实例中选择出满足条件的行。操作符： $\sigma$

投影：从关系实例中抽出所需的一系列或多列。操作符： $\pi$

条件表达式中的比较操作符： $>$ ， $>=$ ， $<$ ， $<=$ ， $=$ ， $\neq$ 。



对应SQL语句的Where子句



对应SQL语句的select子句

## 例1: 查询信息系（IS系）全体学生

$\sigma_{Sdept = 'IS'}(Student)$  或  $\sigma_{5 = 'IS'}(Student)$

结果:

Sno	Sname	Ssex	Sage	Sdept
200215122	刘晨	女	19	IS
200215125	张立	男	19	IS

## 例2: 查询年龄小于20岁的学生

$\sigma_{Sage < 20}(Student)$  或  $\sigma_{4 < 20}(Student)$

结果:

Sno	Sname	Ssex	Sage	Sdept
200215122	刘晨	女	19	IS
200215123	王敏	女	18	MA
200215125	张立	男	19	IS

### 例3: 查询学生的姓名和所在系

即求Student关系上学生姓名和所在系两个属性上的投影

$\pi_{\text{Sname}, \text{Sdept}}(\text{Student})$  或  $\pi_{2, 5}(\text{Student})$

结果:

Sname	Sdept
李勇	CS
刘晨	IS
王敏	MA
张立	IS

### 例4: 查询学生关系Student中都有哪些系

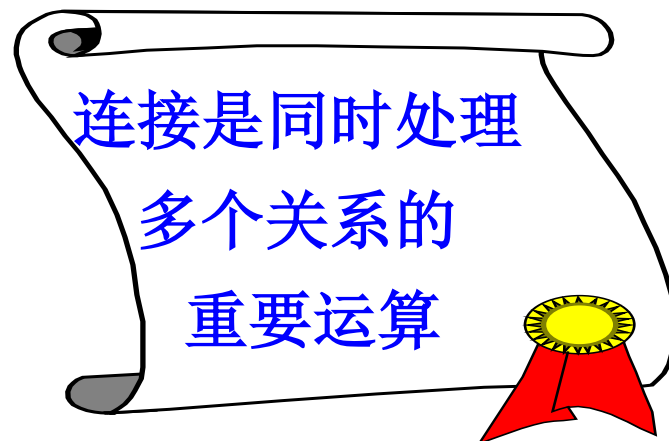
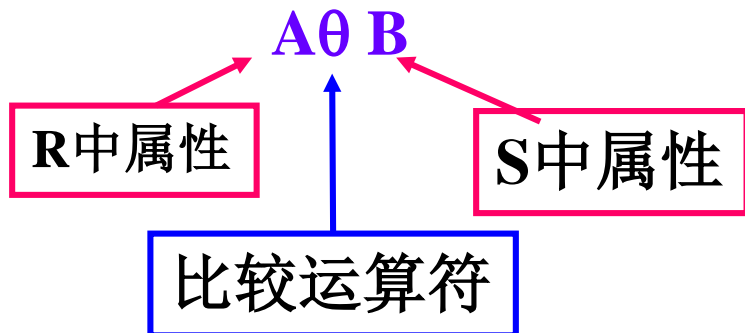
$\pi_{\text{Sdept}}(\text{Student})$

结果:

Sdept
CS
IS
MA

### 3、连接 (Join)

$R \bowtie S$ : 从两个关系的笛卡尔积中选取属性间满足条件  $A \theta B$  的元组。



说明:

◆  $R \bowtie_{A \theta B} S = \sigma_{A \theta B} (R \times S)$

- ◆ 当  $\theta$  为等号且  $A$ 、 $B$  两属性相同时, 称为自然连接, 记作  $R \bowtie S$

- ◆ 自然连接将去掉重复属性

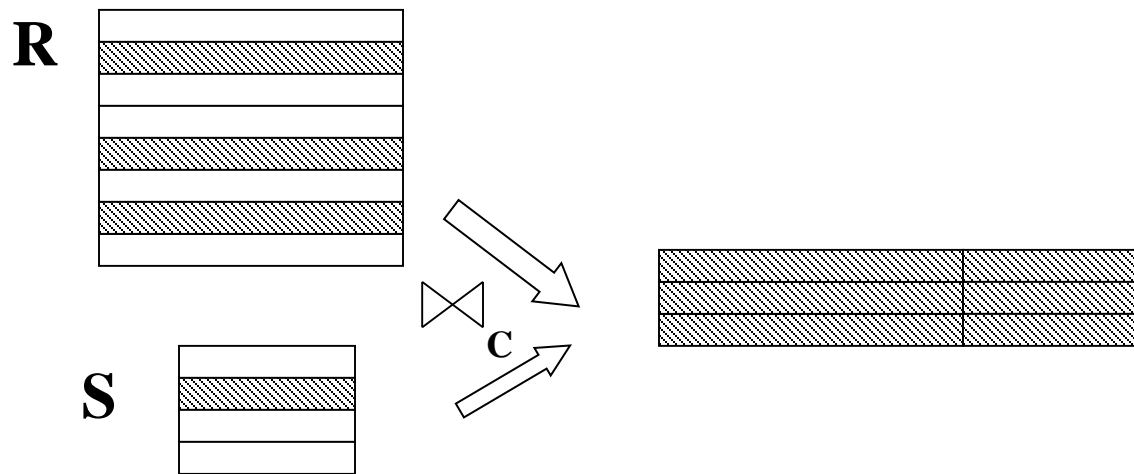
若仅有  $\theta$  为等号的条件, 称为等值连接

种类：条件连接、等值连接、自然连接、外连接。

## (1) Condition Joins(条件连接)

概念： $\mathbf{R} \bowtie_c \mathbf{S} = \sigma_c(\mathbf{R} \times \mathbf{S})$

说明：条件c会用到R和S的属性，如R.name，R.i(位置)。



## 例5: 关系 $R$ 和关系 $S$ 如下所示

$R$

$A$	$B$	$C$
$a_1$	$b_1$	5
$a_1$	$b_2$	6
$a_2$	$b_3$	8
$a_2$	$b_4$	12

$S$

$B$	$E$
$b_1$	3
$b_2$	7
$b_3$	10
$b_3$	2
$b_5$	2

$R \bowtie S$   
 $C < E$

$A$	$R.B$	$C$	$S.B$	$E$
$a_1$	$b_1$	5	$b_2$	7
$a_1$	$b_1$	5	$b_3$	10
$a_1$	$b_2$	6	$b_2$	7
$a_1$	$b_2$	6	$b_3$	10
$a_2$	$b_3$	8	$b_3$	10

## (2)等值连接 (Equijoin)

**概念：**是条件连接的特例，即连接条件由等式组成，如  $R.name1=S.name2$ 。

从关系R与S的广义笛卡尔积中选取A、B属性值相等的那些元组，即等值连接为：

**示例：**  $R \bowtie_{R.B=S.B} S$

R		
A	B	C
$a_1$	$b_1$	5
$a_1$	$b_2$	6
$a_2$	$b_3$	8
$a_2$	$b_4$	12

S	
B	E
$b_1$	3
$b_2$	7
$b_3$	10
$b_3$	2
$b_5$	2

A	R.B	C	S.B	E
$a_1$	$b_1$	5	$b_1$	3
$a_1$	$b_2$	6	$b_2$	7
$a_2$	$b_3$	8	$b_3$	10
$a_2$	$b_3$	8	$b_3$	2



### (3) Natural Join(自然连接)

**概念：**是等连接的特例，即：等式中所涉及的字段名相同，这时可忽略连接条件，即为： $R \bowtie S$ 。

<i>R</i>		
<i>A</i>	<i>B</i>	<i>C</i>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>1</sub>	5
<i>a</i> <sub>1</sub>	<i>b</i> <sub>2</sub>	6
<i>a</i> <sub>2</sub>	<i>b</i> <sub>3</sub>	8
<i>a</i> <sub>2</sub>	<i>b</i> <sub>4</sub>	12

<i>S</i>	
<i>B</i>	<i>E</i>
<i>b</i> <sub>1</sub>	3
<i>b</i> <sub>2</sub>	7
<i>b</i> <sub>3</sub>	10
<i>b</i> <sub>3</sub>	2
<i>b</i> <sub>5</sub>	2

**示例：** $R \bowtie S$

<i>A</i>	<i>B</i>	<i>C</i>	<i>E</i>
<i>a</i> <sub>1</sub>	<i>b</i> <sub>1</sub>	5	3
<i>a</i> <sub>1</sub>	<i>b</i> <sub>2</sub>	6	7
<i>a</i> <sub>2</sub>	<i>b</i> <sub>3</sub>	8	10
<i>a</i> <sub>2</sub>	<i>b</i> <sub>3</sub>	8	2

## 4、除 (Division)

- ◆ 用途：在表达某些查询时有用，例如“查询已选修了所有课程的学生学号”。商并不经常使用，所以，数据库系统并没有将其作为实际操作符来实现。
- ◆ 概念：如存在 $R(X,Y)$ 和 $S(Y,Z)$ 两个关系， $X,Y,Z$ 分别为属性组（如： $X=\{x_1,x_2,\dots,x_n\}$ ，即 $X$ 是由 $n$ 列构成， $Y,Z$ 类似），则 $R \div S$ 结果得到一个新的关系 $P(X)$ ， $P$ 是 $R$ 中满足下列条件的元组在 $X$ 属性列上的投影：元组在 $X$ 上分量值 $x$ 的像集 $Y_x$ 包含 $S$ 在 $Y$ 上投影的集合。

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

其中 $Y_x$ 为 $X$ 在 $R$ 中的象集， $x = tr[X]$ ，即 $x$ 在 $R$ 中存在与 $\Pi_Y(S)$ 中的每个元素对应的记录，除操作是同时从行和列角度进行运算。

示例：设关系订购和零件数据，如表1和表2，求订购÷零件。

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

表1 订购关系 **R**

工程号	工序	零件号
a1	G1	b1
a2	G2	b1
a3	G4	b4
a1	G3	b2
a4	G3	b1
a2	G2	b2
a1	G1	b2

表2 零件关系 **S**

零件号	零件名	颜色
b1	螺母	红色
b2	螺钉	蓝色

X代表工程号和工序，Y代表零件号

某个X对应的Y值

$Y_X$

$\Pi_Y(S)$

$\Pi_X(R)$

订购**R**÷零件结果**S**

工程号	工序
a1	G1
a2	G2
a3	G4
a1	G3
a4	G3

零件号
b1
b2

工程号	工序
a1	G1
a2	G2

## 示例2:

<b>R</b>	<u>sid</u>	<u>cid</u>
	3	101
	3	102
	3	103
	3	104
	5	101
	5	102
	8	102
	10	102
	10	104

	<u>cid</u>
<b>S1</b>	102
<hr/>	
<b>S2</b>	102
	104
<hr/>	
<b>S3</b>	101
	102
	104

	<u>sid</u>
<b>R ÷ S1</b>	3
	5
	8
	10
<hr/>	
<b>R ÷ S2</b>	3
	10
<hr/>	
<b>R ÷ S3</b>	3

用操作符表达:  $\mathbf{R \div S = \Pi_x(R) - \Pi_x((\Pi_x(R) \times \Pi_y(S)) - R)}$

## R ÷ S的具体计算过程如下：

$$\mathbf{R} \div \mathbf{S} = \Pi_{\mathbf{x}}(\mathbf{R}) - \Pi_{\mathbf{x}}((\Pi_{\mathbf{x}}(\mathbf{R}) \times \Pi_{\mathbf{Y}}(\mathbf{S})) - \mathbf{R})$$

- ① 找出关系R和关系S中相同的属性，即Y属性。在关系S中对Y做投影（即 $\Pi_{\mathbf{Y}}(\mathbf{S})$ ）；
- ② 设被除关系R与S的不相同的列为X，对关系R在X上做消除重复值的投影（即： $\Pi_{\mathbf{x}}(\mathbf{R})$ ）；
- ③ 对①②步求出的关系做笛卡尔积： $\Pi_{\mathbf{x}}(\mathbf{R}) \times \Pi_{\mathbf{Y}}(\mathbf{S})$
- ④ 对③步的结果与R做差： $(\Pi_{\mathbf{x}}(\mathbf{R}) \times \Pi_{\mathbf{Y}}(\mathbf{S})) - \mathbf{R}$
- ⑤ 对④步的结果做投影 $\Pi_{\mathbf{x}}((\Pi_{\mathbf{x}}(\mathbf{R}) \times \Pi_{\mathbf{Y}}(\mathbf{S})) - \mathbf{R})$
- ⑥  $\mathbf{R} \div \mathbf{S}$ 就是②-⑤。

示例：计算订购÷零件的过程

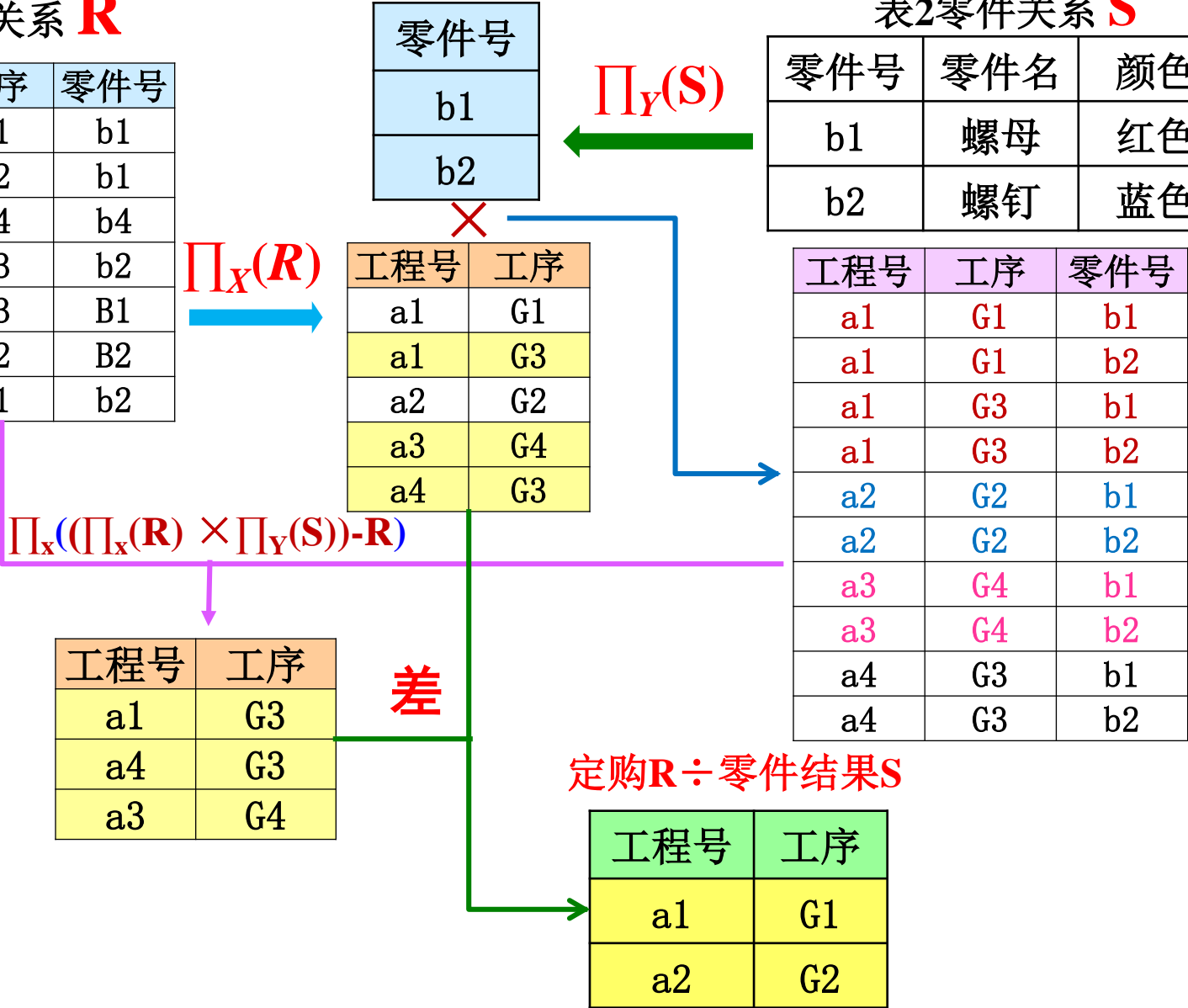
$$R \div S = \Pi_x(R) - \Pi_x((\Pi_x(R) \times \Pi_y(S)) - R)$$

表1 订购关系 **R**

工程号	工序	零件号
a1	G1	b1
a2	G2	b1
a3	G4	b4
a1	G3	b2
a4	G3	B1
a2	G2	B2
a1	G1	b2

表2 零件关系 **S**

零件号	零件名	颜色
b1	螺母	红色
b2	螺钉	蓝色



订购**R**÷零件结果**S**

## 5、外连接(Outer Joins)

**概念：** 涉及有空值的自然连接，是自然连接的特例。

**说明：** 自然连接是寻找相同字段值相等的行。但如果一个关系中的该字段在另一关系中没有相等值的行，自然连接不会显示该行，而外连接则将以NULL值形式显示该行。

**外连接的种类：**

- ① 左外连接 (LEFT OUTER JOIN)
- ② 右外连接 (RIGHT OUTER JOIN)
- ③ 全外连接 (FULL OUTER JOIN)

**说明：**

① 与外连接对应，前面三种连接为**内连接(Inner Join)**；

② 关系代数中没有外连接的描述，但SQL标准中有相应的三种外连接查询语句；

**左外连接：**对于 $R \bowtie S$ ，如果在S中没有匹配R的行，则以NULL值表示，最后的结果是以左边的关系R为准，即左边关系中的所有行均应出现在结果中，如果在S中没有对应的行，则以NULL表示之。

<b>R</b>	<u>sid</u>	sname	age	grade	<b>S</b>	<u>sid</u>	cid	score	<b>结果：</b>	<u>sid</u>	cid
	8	何大明	19	2		8	101	91		8	101
	11	李 峰	20	3		35	106	84		11	null
	35	陈 胜	21	4						35	106

左外连接示意图



**右外连接：**对于  $R \bowtie S$ ，如果在  $R$  中没有匹配  $S$  的行，则以 **NULL** 值表示，最后的结果以右边的关系  $S$  为准。

<b>R</b>	<u>sid</u>	sname	age	grade	<b>S</b>	<u>sid</u>	cid	score	<b>结果：</b>	<u>sid</u>	cid	sname
	8	何大明	19	2		8	101	91		8	101	何大明
	11	李 峰	20	3		35	106	84		35	106	陈 胜
	35	陈 胜	21	4		66	119	88		66	119	null

右外连接示意图

全外连接：对于R⋈S，没有匹配的R和S的行，也都出现于结果中。

R	sid	sname	age	grade	S	sid	cid	score	结果:	sid	cid	sname
	8	何大明	19	2		8	101	91		8	101	何大明
	11	李 峰	20	3		35	106	84		11	null	李 峰
	35	陈 胜	21	4		66	119	88		35	106	陈 胜
										66	119	null

全外连接示意图

## 请大家课后探讨下列关系模型的关系运算

选课系统的数据库表格如下：课程信息表（Course）、教师信息表（Teacher）、开课计划表（Plan）、学生信息表（Student）、选课注册表（Register）、学院信息表（College）组成。（其具体定义见教材）。

针对下列问题，如何应用关系模型运算方法进行数据处理：

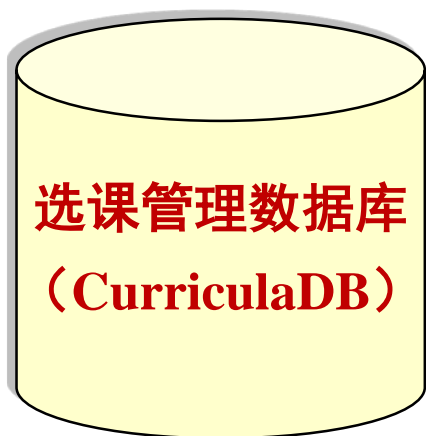
- 1、如何查找计算机专业的学生？
- 2、如何查找计算机专业并且年龄大于20岁的学生？
- 3、如何查找教师的姓名、职称、所在学院信息？
- 4、如何查找教师的姓名和开的课的名称？
- 5、如何查找学院计划开出的课程清单（学院名称，课程名称）？
- 6、如何检索学习课程号为C2的学生学号和成绩？
- 7、如何检索学习课程号为C2的学生学号和姓名？
- 8、如何检索选修课程名为MATHS的学生学号和姓名？
- 9、如何检索选修课程号为C2或C4的学生学号？
- 10、如何检索至少选修课程号为C2或C4的学生学号？
- 11、如何检索不学C2课的学生姓名、年龄？
- 12、如何检索学习全部课程的学生姓名？

# PostgreSQL数据库关系操作实践

- 掌握创建PostgreSQL关系数据库方法
- 掌握在PostgreSQL数据库中创建关系表方法
- 掌握在PostgreSQL数据库中定义关系表的主键、代理键与外键方法
- 掌握在PostgreSQL数据库中定义关系表的实体完整性、参照完整性、用户自定义完整性方法

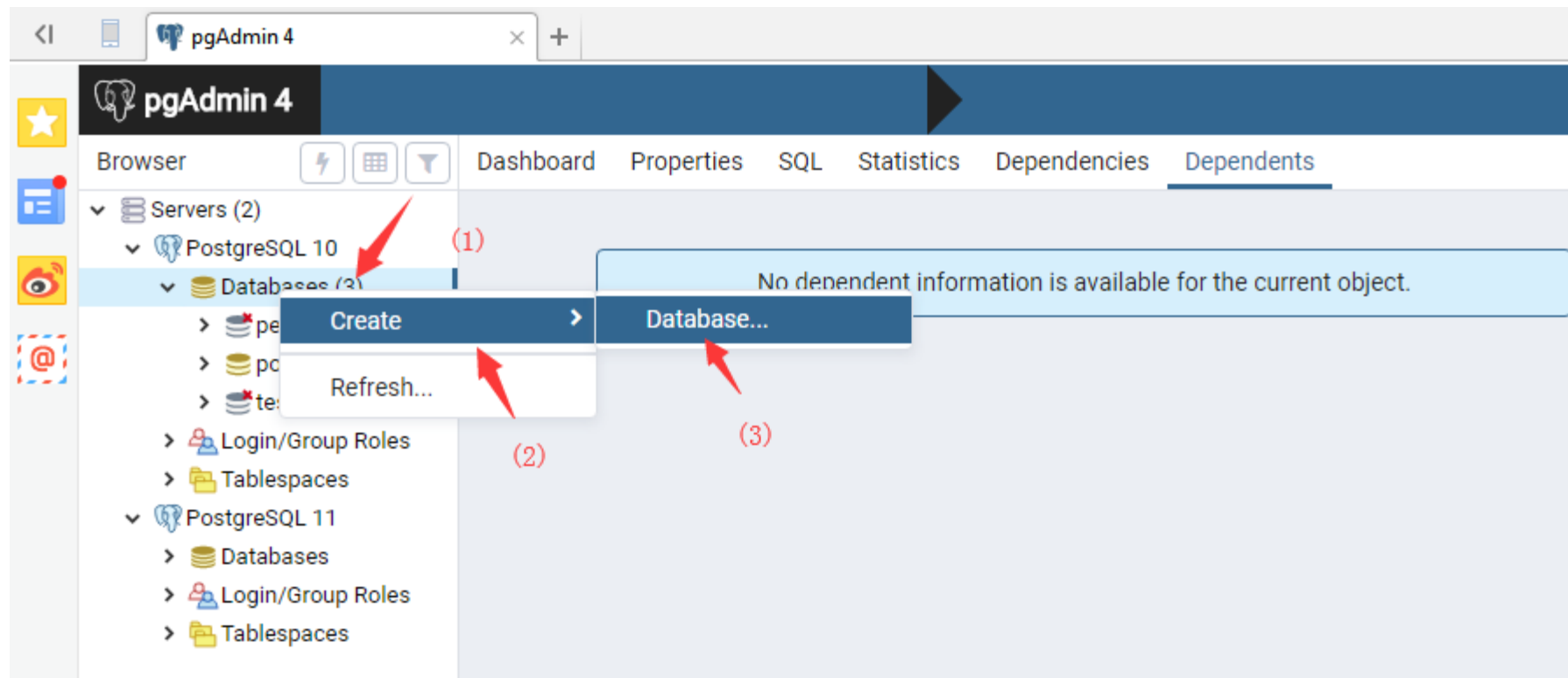
# 一、项目案例——选课管理系统数据库关系表实践

本节将围绕“选课管理系统”项目案例，在PostgreSQL数据库中创建关系表及其完整性约束，并理解本章所学习的关系模型基本概念和关系操作原理。

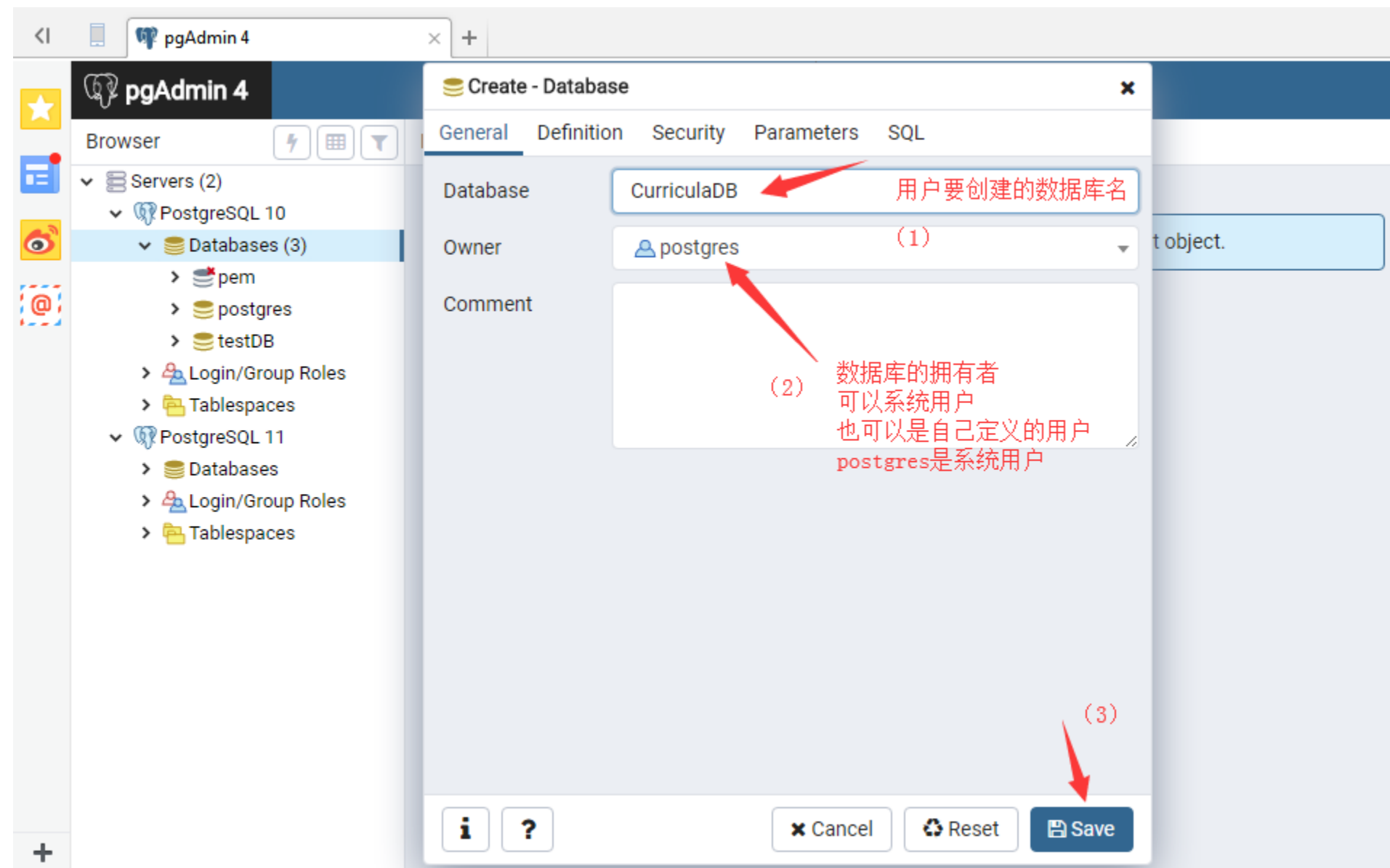


- 课程表 (COURSE)
- 教师表 (TEACHER)
- 开课计划表 (PLAN)
- 学生表 (STUDENT)
- 选课注册表 (REGISTER)
- 学院信息表 (COLLEGE)

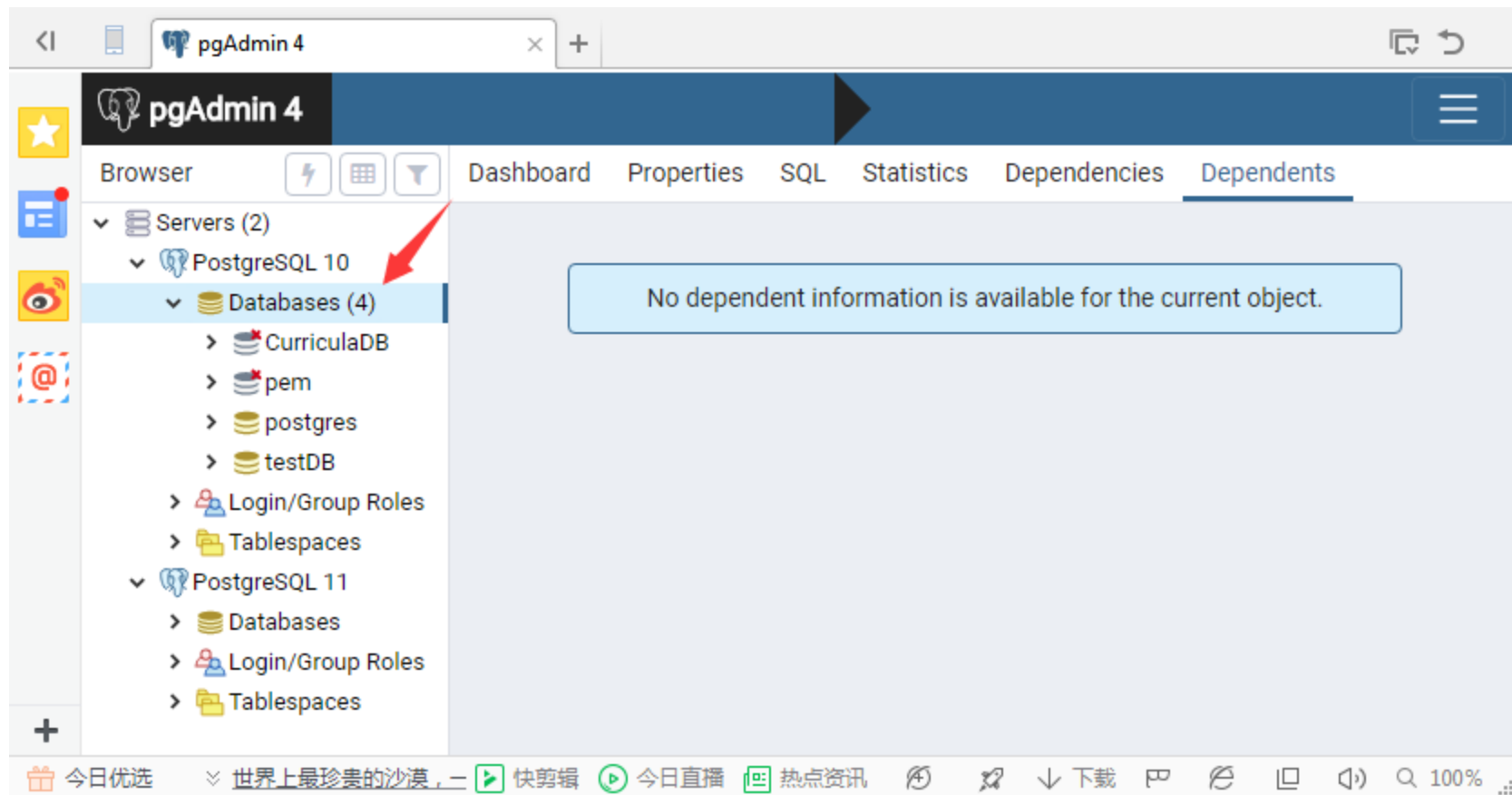
# (1) 使用pgAdmin4创建数据库



# 指定数据库名和拥有者



# 数据库创建成功





## 课程表（COURSE）

字段名称	字段编码	数据类型	字段大小	必填字段	是否为键
课程编号	CourseID	文本	4	是	主键
课程名	CourseName	文本	20	是	否
课程类别	CourseType	文本	10	否	否
学分	CourseCredit	数字	短整型	否	否
学时	CoursePeriod	数字	短整型	否	否
考核方式	TestMethod	文本	10	否	否

## 教师表（TEACHER）

字段名称	字段编码	数据类型	字段大小	必填字段	是否为键
教师编号	TeacherID	文本	4	是	主键
姓名	TeacherName	文本	10	是	否
性别	TeacherGender	文本	2	否	否
职称	TeacherTitle	文本	6	否	否
所属学院	CollegeID	文本	3	否	外键
联系电话	TeacherPhone	文本	11	否	否

## 开课计划表 (PLAN)

字段名称	字段编码	数据类型	字段大小	必填字段	是否为键
开课编号	CoursePlanID	自动编号	长整型	是	代理键
课程编号	CourseID	文本	4	是	外键
教师编号	TeacherID	文本	4	是	外键
地点	CourseRoom	文本	30	否	否
时间	CourseTime	文本	30	否	否
备注	Note	文本	50	否	否

## 学生表 (STUDENT)

字段名称	字段编码	数据类型	字段大小	必填字段	是否为键
学号	<b>StudentID</b>	文本	<b>13</b>	是	主键
姓名	<b>StudentName</b>	文本	<b>10</b>	是	否
性别	<b>StudentGender</b>	文本	<b>2</b>	否	否
出生日期	<b>BirthDay</b>	日期	短日期	否	否
专业	<b>Major</b>	文本	<b>30</b>	否	否
手机号	<b>StudentPhone</b>	文本	<b>11</b>	否	否

## 选课注册表 (REGISTER)

字段名称	字段编码	数据类型	字段大小	必填字段	是否为键
注册编号	CourseRegID	自动编号	长整型	是	代理键
开课编号	CoursePlanID	数字	长整型	是	外键
学号	StudentID	文本	13	是	外键
备注	Note	文本	30	否	否

## 学院信息表（COLLEGE）

字段名称	字段编码	数据类型	字段大小	必填字段	是否为键
学院编号	CollegeID	文本	3	是	主键
学院名称	CollegeName	文本	40	是	否
学院介绍	CollegeIntro	文本	200	否	否
学院电话	CollegeTel	文本	30	否	否

# 使用pgAdmin 4创建PostgreSQL数据库表

The screenshot shows the pgAdmin 4 interface with the following elements:

- Browser Panel (Left):** Displays a tree structure of the database. The 'CurriculaDB' database is selected, and the 'public' schema is expanded. The 'Tables' option is highlighted in the 'Create' menu.
- Main Panel (Right):** Displays a table with columns 'Type', 'Name', and 'Restriction'. The message 'No data found' is shown.
- Annotations:** Red arrows and text labels indicate the steps: (1) 连接数据库 (Connect to database), (2) 打开模式 (Open schema), (3) 创建 (Create), and (4) 创建表 (Create table).

**Browser Panel Tree Structure:**

- Servers (2)
  - PostgreSQL 10
    - Databases (4)
      - CurriculaDB** (Selected)
      - Casts
      - Catalogs
      - Event Triggers
      - Extensions
      - Foreign Data Wrappers
      - Languages
      - Schemas (1)** (Expanded)
        - public** (Expanded)
          - Collations
          - Domains
          - FTS Configurations
          - FTS Dictionaries
          - FTS Parsers
          - FTS Templates
          - Foreign Tables
          - Function Groups
          - Materialized Views
          - Sequences
          - Tables** (Selected)
          - Trigger Functions
          - Types
          - Views

**Main Panel Table:**

Type	Name	Restriction
No data found		

## 使用pgAdmin 4创建学院信息表（COLLEGE）

The image shows the 'Create - Table' dialog box in pgAdmin 4. The 'General' tab is selected and highlighted with a red box. A red arrow points from the 'General' tab to the 'Name' field, which contains the text 'COLLEGE' and is also highlighted with a red box. Another red arrow points from the 'Name' field to the 'Comment' field, which contains the text '学院信息表（COLLEGE）' and is also highlighted with a red box. The 'Owner' field is set to 'postgres', the 'Schema' field is set to 'public', and the 'Partitioned Table?' checkbox is unchecked. The 'Tablespace' field is set to 'Select from the list'. At the bottom, there are buttons for 'Cancel', 'Reset', and 'Save'.

Create - Table

General Columns Constraints Advanced Partition Parameter Security SQL

Name COLLEGE

Owner postgres

Schema public

Tablespace Select from the list

Partitioned Table? No

Comment 学院信息表（COLLEGE）

Cancel Reset Save



# 使用pgAdmin 4创建学院信息表（COLLEGE）续

Create - Table

General Columns Constraints Advanced Partition Parameter Security SQL

Inherited from table(s) Select to inherit from...

Columns

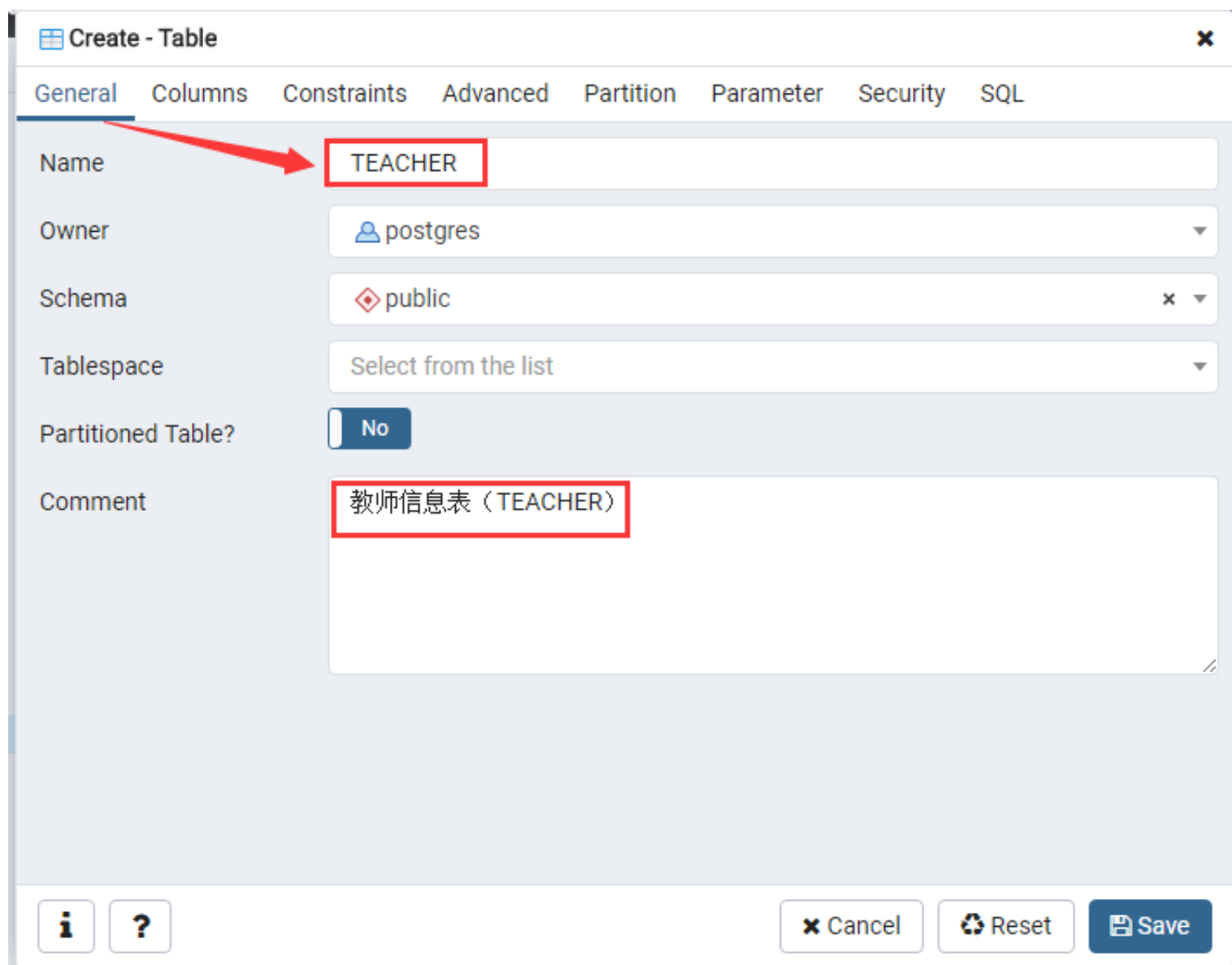
(2) 添加数据库表的列信息

	Name	Data type	Length	Precision	Not NULL?	Primary key?
		CollegelD	character	5	<input checked="" type="checkbox"/> Yes	<input checked="" type="checkbox"/> Yes
		CollegeName	character	30	<input checked="" type="checkbox"/> Yes	<input type="checkbox"/> No
		CollegelIntro	character varying	300	<input type="checkbox"/> No	<input type="checkbox"/> No
		CollegeTel	character	13	<input type="checkbox"/> No	<input type="checkbox"/> No

(3) 保存

Cancel Reset Save

# 使用pgAdmin 4创建教师信息表（TEACHER）



The image shows the 'Create - Table' dialog box in pgAdmin 4. The 'General' tab is selected. The 'Name' field is set to 'TEACHER', the 'Owner' is 'postgres', and the 'Schema' is 'public'. The 'Partitioned Table?' checkbox is unchecked. The 'Comment' field contains '教师信息表 (TEACHER)'. A red arrow points to the 'Name' field, and red boxes highlight the 'Name' and 'Comment' fields. The bottom of the dialog has buttons for 'Cancel', 'Reset', and 'Save'.

Field	Value
Name	TEACHER
Owner	postgres
Schema	public
Tablespace	Select from the list
Partitioned Table?	No
Comment	教师信息表 (TEACHER)

# 使用pgAdmin 4创建教师信息表（TEACHER）续

Create - Table

General

Columns

Constraints

Advanced

Partition

Parameter













Security

SQL

Inherited from table(s) 

Select to inherit from...

Columns

	Name	Data type	Length	Precision	Not NULL?	Primary key?
 	TeacherID	<div>character</div>	8		<div>Yes</div>	<div>Yes</div>
 	TeacherName	<div>character</div>	20		<div>Yes</div>	<div>No</div>
 	TeacherGender	<div>character</div>	6		<div>No</div>	<div>No</div>
 	TeacherITitle	<div>character</div>	8		<div>No</div>	<div>No</div>
 	CollegeID	<div>character</div>	5		<div>No</div>	<div>No</div>
 	TeacherPhone	<div>character</div>	13		<div>No</div>	<div>No</div>

i

?

Cancel

Reset

Save

# 使用pgAdmin 4给教师信息表（TEACHER）创建外键约束

The screenshot shows the pgAdmin 4 interface for creating a foreign key constraint on the TEACHER table. The 'Constraints' tab is active, and the 'Foreign Key' sub-tab is selected. A new foreign key named 'fk-teacherCollegeID' is being created. The 'Columns' sub-tab is also shown, where 'CollegeID' is selected as the local column, 'public.\"COLLEGE\"' is selected as the referenced table, and 'CollegeID' is selected as the referenced column. The 'Save' button is highlighted.

**TEACHER**

General Columns **Constraints** Advanced Parameter Security SQL

Primary Key **Foreign Key** Check Unique Exclude

Foreign key

Name **fk-teacherCollegeID** Columns (CollegeID) -> (CollegeID)

General Definition **Columns** Action

Columns

Local column CollegeID

References public.\"COLLEGE\"

Referencing CollegeID

Local CollegeID Referenced CollegeID

Cancel Reset **Save**

(1) 给外键命名

(2) 选择外键列

(3) 选择主表

(4) 选择参照的主表中的主键

(5) 保存

# 使用pgAdmin 4给教师信息表（TEACHER）创建外键约束的处理

TEACHER

GeneralColumnsConstraintsAdvancedParameterSecuritySQL

Primary KeyForeign KeyCheckUniqueExclude

Foreign key

+

Name	Columns
fk-teacherCollegeID	(CollegeID) -> (CollegeID)

GeneralDefinitionColumnsAction

On update

CASCADE

On delete

CASCADE

NO ACTION

RESTRICT

CASCADE

SET NULL

SET DEFAULT

(1) 在主表college执行更新操作时, 在从表teacher表如何处理

(2) 定义在主表college表上执行删除操作, 从表teacher表如何处理

Cancel

Reset

Save

# 使用pgAdmin 4给教师信息表（TEACHER）创建check约束

