复习

- 选择题（1’*20)
- 简答题（5’*3)
- 规范化(15’)
- SQL+关系代数（40’)
- ER，数据库设计(10’)

数据库

- DB，DBMS，DBS
- DBMS功能
- DDL,DML,DCL

DBMS的主要功能（1）

- 数据定义功能
  - 提供数据定义语言DDL（Data Definition Language）
  - 定义数据库中的数据对象（在关系数据库管理系统中就是指定义数据库、表、视图、索引等）
  - 还包括定义完整性约束和保密限制等

DBMS的主要功能（2）

- 数据操纵功能
  - 提供数据操纵语言DML（Data Manipulation Language）操纵数据
  - 实现对数据库的基本操作——查询、插入、删除和修改

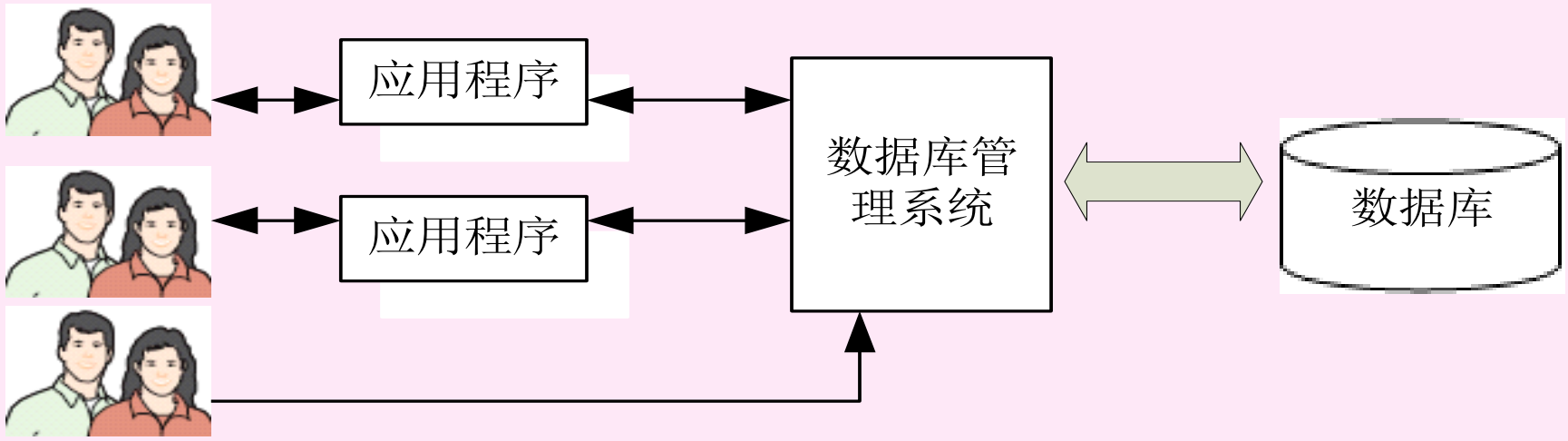
DBMS的主要功能（2）

- 数据库的运行管理
  - 安全性控制
  - 完整性控制
  - 多用户对数据的并发使用时的并发控制

DBMS的主要功能（3）

- 数据库的建立和维护功能(实用程序)
  - 数据库数据批量装载
  - 数据库转储
  - 介质故障恢复
  - 数据库的重组织
  - 性能监视、分析等

数据库系统构成图



数据库系统构成（1）

- 数据库
  - 数据库是数据库系统的基石。
  - 数据库按一定的数据模型组织、描述和存储数据，具有较小的冗余度、较高的数据独立性和易扩展性，并可被多个用户共享使用。

数据库系统构成（2）

- 数据库管理系统
  - 数据库系统的核心
  - 负责数据库存取、维护和管理的系统软件
  - 其功能的强弱是衡量数据库系统性能优劣的主要指标
  - 数据库中的数据由数据库管理系统进行统一管理和控制
  - 用户对数据库进行的各种操作是由数据库管理系统实现的

数据库系统构成（3）

- 应用程序
  - 在数据库管理系统的基础上，由用户根据应用的实际需要开发的、处理特定业务的应用程序。
  - 普通用户主要利用应用程序来与数据库管理系统交互，以存取数据库内的数据。

数据库系统构成（4）

- 用户
  - 管理、开发、使用数据库系统的所有人员
  - 包括数据库管理员、应用程序员和终端用户
  - 数据库管理人员可以直接操纵数据库管理系统。数据库的实施、管理、维护一般都由数据库管理人员来完成。
  - **DBA**（Database Administrator）——数据库管理员
    - 负责整个数据库系统的建立、管理、维护、协调工作的专门人员

数据库技术的发展过程

- 数据管理技术的发展过程

  在应用需求的推动下，在计算机硬件、软件发展的基础上，数据管理技术经历了下面三个阶段：

  - 人工管理阶段(40年代中--50年代中)
  - 文件系统阶段(50年代末--60年代中)
  - 数据库系统阶段(60年代末--现在)

数据库技术的发展过程

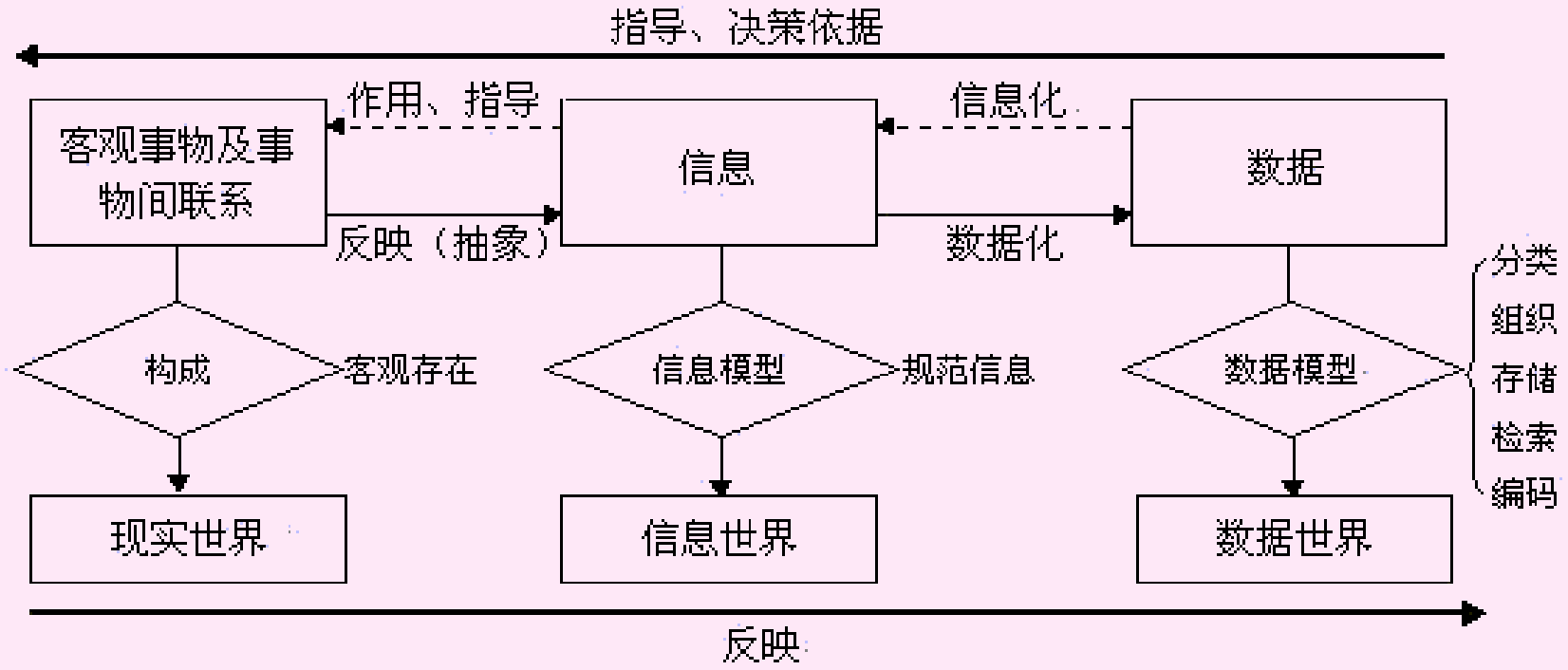
- 数据管理技术的发展过程

  在应用需求的推动下，在计算机硬件、软件发展的基础上，数据管理技术经历了下面三个阶段：

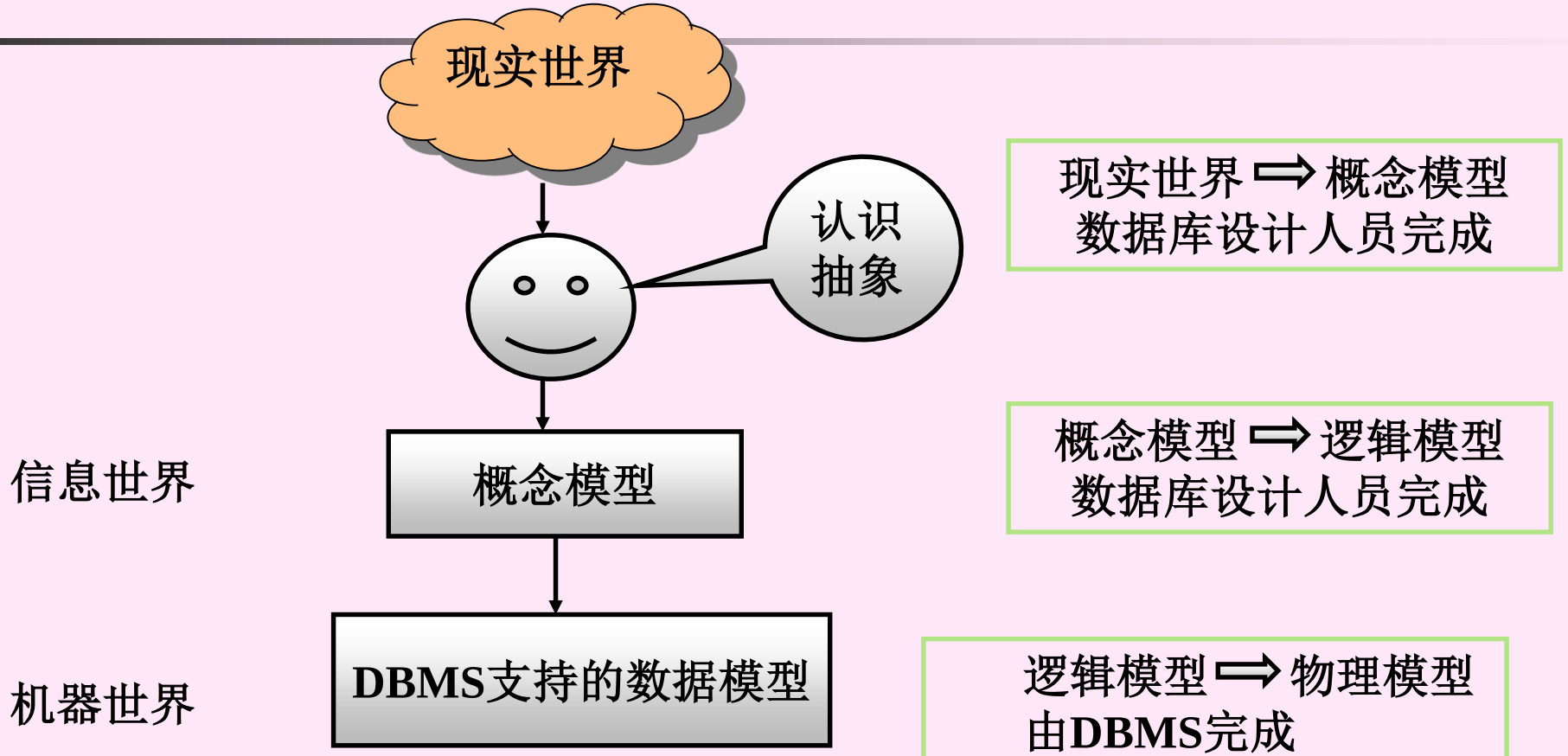
  - 人工管理阶段(40年代中--50年代中)
  - 文件系统阶段(50年代末--60年代中)
  - 数据库系统阶段(60年代末--现在)

数据模型

信息的三个世界


指导、决策依据
作用、指导
信息化
客观事物及事物间联系
信息
数据
反映（抽象）
数据化
构成
客观存在
信息模型
规范信息
数据模型
分类
组织
存储
检索
编码
现实世界
信息世界
数据世界
反映

客观对象的抽象过程（续）



现实世界中客观对象的抽象过程

数据模型的组成要素

- 数据结构
- 数据操作
- 数据的完整性约束条件

数据结构

- 什么是数据结构
  - 数据库的组成对象以及对象之间的联系
- 两类对象
  - 与数据类型、内容、性质有关的对象（对数据的描述）
  - 与数据之间联系有关的对象（对数据间联系的描述）
- 数据结构是对系统静态特性的描述
- 数据结构是刻画一个数据模型性质最重要的方面，因此在数据库系统中通常按照数据结构的类型来命名数据模型
  - 采用层次型、网状型和关系型数据结构的数据模型分别被称为层次模型、网状模型和关系模型

数据操作

- 数据操作
  - 对数据库中各种对象的实例允许执行的操作及有关的操作规则。
- 数据操作的类型
  - 检索和更新（包括插入、删除、修改）
- 数据操作是对系统动态特性的描述。
- 数据模型要给出这些操作确切的含义、操作规则和实现操作的语言。

数据的约束条件

- 数据的完整性约束条件
  - 一组完整性规则的集合。
    - 层次模型中的每个结点有且仅有一个双亲结点；关系模型中的实体完整性和参照完整性
  - 完整性规则是给定的数据模型中数据及其联系所具有的制约和依存规则，用以限定符合数据模型的数据库状态以及状态的变化，以保证数据的正确、有效、相容。
    - 数据约束还应包括能够反映某一应用所涉及的数据必须遵守的特定的语义约束条件

常用数据模型

- 非关系模型
  - 层次模型(Hierarchical Model)
  - 网状模型(Network Model)
- **关系模型(Relational Model)**
- **面向对象模型(Object Oriented Model）**

关系数据模型的数据结构

- 在用户观点下，关系模型中数据的逻辑结构是一张二维表，它由行和列组成。

学生登记表

属性

元组

学 号	姓 名	年 龄	性 别	系 名	年 级
2005004	王小明	19	女	社会学	2005
2005006	黄大鹏	20	男	商品学	2005
2005008	张文斌	18	女	法律	2005
…	…	…	…	…	…

关系模型的基本概念

- **关系（Relation）**

  一个关系对应通常说的一张表。

- **元组（Tuple）**

  表中的一行即为一个元组。

- **属性（Attribute）**

  表中的一列即为一个属性，给每一个属性起一个名称即属性名。

关系模型的基本概念

- **码（Key）**

  表中的某个属性（组），它可以唯一确定一个元组。

- **域（Domain）**

  属性的取值范围。

- **分量**

  元组中的一个属性值。

- **关系模式**

  对关系的描述

  关系名（属性1，属性2，…，属性n）

  学生（学号，姓名，年龄，性别，系，年级）

关系数据模型的数据结构(续）

- 实体及实体间的联系的表示方法
  - 实体型：直接用关系（表）表示。
  - 属性：用属性名表示。
  - 联系：用键表示。
- 在层次和网状模型中，联系用指针来实现，在关系模型中，记录之间的联系通过键来实现
- 关系模型的操作语言是SQL

关系数据模型的数据结构（续）

- 关系必须是规范化的，满足一定的规范条件

  最基本的规范条件：关系的每一个分量必须是一个不可分的数据项，

  不允许表中还有表

  图1.27中工资和扣除是可分的数据项，不符合关系模型要求

职工号	姓名	职 称	工 资			扣 除		实 发
			基 本	津 贴	职务	房 租	水 电	
86051	陈 平	讲 师	1305	1200	50	160	112	2283

图1.27 一个工资表(表中有表)实例

关系数据模型的操纵与完整性约束

- 数据操作是集合操作，操作对象和操作结果都是关系
  - 查询 /插入 /删除 /更新
- 数据操作是集合操作，操作对象和操作结果都是关系，即若干元组的集合
- 存取路径对用户隐蔽，用户只要指出“干什么”，不必详细说明“怎么干”

关系模型的基本概念

- 关系模型的组成
  - 关系数据结构
  - 关系操作集合
  - 关系完整性约束

1. 在关系模型中，数据是以二维表的形式存在的，这个二维表就叫做关系

2.关系理论是以集合代数理论为基础的，因此，我们可以用集合代数给出二维表的“关系”定义

关系模型的基本概念

- 关系模型的组成
  - 关系数据结构
  - 关系操作集合
  - 关系完整性约束

1. 关系代数

用代数的方法表示关系模型

2.关系演算

用逻辑方法表示关系模型

3、SQL

具有关系代数和关系演算双重特点的语言

关系模型的基本概念

- 关系模型的组成
  - 关系数据结构
  - 关系操作集合
  - 关系完整性约束

1. 实体完整性
2. 参照完整性
3. 用户定义的完整性

关系的数学定义

- 【定义2】关系 $D_1 \times D_2 \times \ldots \times D_n$ 的子集叫作在域 D_1，D_2，…，D_n 上的关系，表示为 $R(D_1, D_2, \ldots, D_n)$
  - R：关系名
  - n：关系的目或度（Degree）
    - 当n=1时，称为单元关系。
    - 当n=2时，称为二元关系。
    - …
    - 当n=n时，称为n元关系。

D1×D2笛卡尔积的子集

学生关系

speciality	postgraduate
计算机应用	张三
计算机应用	王五
信息管理	李四

Keys（码或键）

- Let K ⊆ R
- *K* is a *superkey*（超码）of *R* if values for *K* are sufficient to identify a unique tuple of each possible relation *r(R)* by "possible *r*" we mean a relation *r* that could exist in the enterprise we are modeling.
  Example: {*customer-name, customer-street*} and
  {*customer-name*}
  are both superkeys of *Customer*, if no two customers can possibly have the same name.
- *K* is a *candidate key*（候选码） if *K* is minimal
  Example: {*customer-name*} is a candidate key for *Customer*, since it is a superkey {assuming no two customers can possibly have the same name), and no subset of it is a superkey.

关系的键（码）

- **候选码（Candidate key）：若关系中的某一属性组的值能唯一地标识一个元组，则称该属性组为候选码**
  - 简单的情况：候选码只包含一个属性
  - “学生关系”中的学号能唯一标识每一个学生，则属性学号是学生关系的候选键。
  - 在“选课关系”中，只有属性的组合“学号+课程号”才能唯一地区分每一条选课记录，则属性集“学号+课程号”是选课关系的候选键。

关系的键（码）

下面给出候选键的形式化定义：

- **设关系R有属性A1，A2，……An，其属性集K=（Ai，Aj，……Ak），当且仅当满足下列条件时，K被称为候选键：**
  - 1. 唯一性（Uniqueness）：关系R的任意两个不同元组，其属性集K的值是不同的。
  - 2.最小性（Minimally）：组成关系键的属性集（Ai，Aj，……Ak）中，任一属性都不能从属性集K中删掉，否则将破坏唯一性的性质

主键

- 如果一个关系中有多个候选键，可以从中选择一个作为查询、插入或删除元组的操作变量，被选用的候选键称为主关系键(Primary Key)，或简称为主键、主码、关系键、关键字。
  - 例如，假设在学生关系中没有重名的学生，则“学号”和“姓名”都可作为学生关系的候选键。如果选定“学号”作为数据操作的依据，则“学号”为主关系键。
- 在关系模式中表示主键
  - 学生（学号，姓名，性别，年龄，系别）

主属性与非码属性

- **主属性（Prime Attribute）：包含在候选码中的的各属性称为主属性。**
- **非码属性（Non-Prime Attribute）：不包含在任何候选码中的属性称为非码属性。**
  - 在最简单的情况下，一个候选码只包含一个属性，如学生关系中的“学号”，教师关系中的“教师号”。
  - 最极端情况，全码关系中所有属性都是主属性

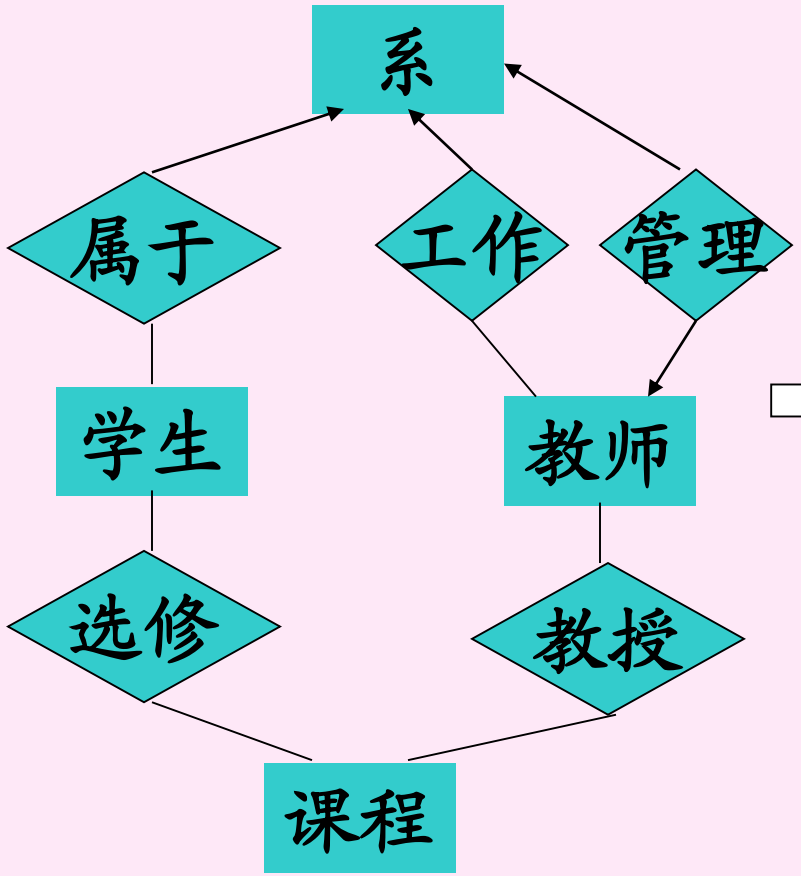
外部关系键

- 如果关系**R2**的一个或一组属性**X**不是**R2**的主码，而是另一关系**R1**的主码，则该属性或属性组**X**称为关系**R2**的外部关系键或外码（**Foreign key**）。

R1	
depid	depname
1	计算机
2	外语系
3	电子系

R2		
id	name	depid
2001	张三	1
2002	李四	1
2003	王五	2
2004	钱六	3

Example



DEPT(D# , DN , DEAN)

S(S# , SN , SEX , AGE , D#)

C(C# , CN , PC# , CREDIT)

SC(S# , C# , SCORE)

PROF(P# , PN, D# , SAL)

TEACH(P# , C#)

1. 实体完整性(Entity Integrity)

- 实体完整性是指主关系键的值不能为空或部分为空。
- **若属性A是基本关系R的主属性，则属性A不能取空值**

实体完整性的说明:

- 现实世界中的实体是可区分的，即它们具有某种唯一性标识。
- 关系模型中以主关系键来唯一标识元组。
  - 例如，学生关系中的属性“学号”可以唯一标识一个元组，也可以唯一标识学生实体。
- 如果主关系键中的值为空或部分为空，即主属性为空，则不符合关系键的定义条件，不能唯一标识元组及与其相对应的实体。
- 主关系键的值不能为空或部分为空

参照完整性(Referential integrity)

- 形式定义如下：
  - 如果属性集K是关系模式R1的主键，K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。
  - 这条规则的实质是“不允许引用不存在的实体”。
- 关系模式R1的关系称为“参照关系”，关系模式R2的关系称为“依赖关系”。
- “主表”和“副表”，“父表”和“子表”。

外键的说明

- 关系R和S不一定是不同的关系
- 目标关系S的主码Ks和参照关系的外码F必须定义在同一个（或一组）域上
- 外码并不一定要与相应的主码同名
- 当外码与相应的主码属于不同关系时，往往取相同的名字，以便于识别

参照完整性

- 在关系数据库中有下列两个关系模式：

  S（S#，SNAME，AGE，SEX）

  SC（S#，C#，GRADE）

- 如果关系SC中有一个元组（S7,C4,80），而学号S7却在关系S中找不到
- 在关系SC中引用了一个不存在的学生实体，这就违反了参照完整性规则。

3. 用户定义完整性（User-defined Integrity）

- 用户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。
  - 例如，选课关系中成绩不能为负数；某些数据的输入格式要有一些限制等
- 关系模型应该提供定义和检验这类完整性的机制，以便用统一的、系统的方法处理它们，而不要由应用程序承担这一功能。

用户定义完整性

例如学生的年龄定义为两位整数，范围还太大，我们可以写如下规则把年龄限制在15～30岁之间：

CHECK（AGE　BETWEEN　15　AND　30）

关系操作

- 关系操作采用集合操作方式，即操作的对象和结构都是集合。
- 常用的关系操作
  - 查询：选择、投影、连接、除、并、交、差
  - 数据更新：插入、删除、修改
  - 查询的表达能力是其中最主要的部分
  - 选择、投影、并、差、笛卡尔积是5种基本操作
- 关系操作的特点
  - 集合操作方式：操作的对象和结果都是集合，一次一集合的方式

关系操作语言

- 关系代数语言
  - 用对关系的运算来表达查询要求
  - 代表：ISBL
- 关系演算语言：用谓词来表达查询要求
  - 元组关系演算语言
    - 谓词变元的基本对象是元组变量
    - 代表：APLHA, QUEL
  - 域关系演算语言
    - 谓词变元的基本对象是域变量
    - 代表：QBE
- 具有关系代数和关系演算双重特点的语言
  - 代表：SQL（Structured Query Language）

关系代数的分类

- 传统的集合运算：把关系看成元组的集合，以元组作为集合中元素来进行运算，其运算是从关系的“水平”方向即行的角度进行的。包括并、差、交和笛卡尔积等运算。
- 专门的关系运算：不仅涉及行运算，也涉及列运算，这种运算是为数据库的应用而引进的特殊运算。包括选取、投影、连接和除法等运算。

关系代数操作符

关系代数运算符

运算符		含义	运算符		含义
集合运算符	∪	并	比较运算符	>	大于
	-	差		≥	大于等于
	∩	交		<	小于
	×	笛卡尔积		≤	小于等于
				=	等于
				<>	不等于

关系代数操作符（续）

关系代数运算符（续）

运算符	含义		运算符	含义	
专门的关系运算符	σ	选择	逻辑运算符	¬	非
	π	投影		∧	与
	⋈	连接		∨	或
	÷	除			

传统的集合运算

对两个关系的集合运算传统的集合运算是二目运算，是在两个关系中进行的。但是并不是任意的两个关系都能进行这种集合运算，而是要在两个满足一定条件的关系中进行运算。

- 相容性定义：设给定两个关系R、S，若满足：

（1） 具有相同的度n;

（2） R中第i个属性和S中第i个属性必须来自同一个域。

则说关系R、S是相容的。除笛卡尔积外，要求参加运算的关系必须满足上述的相容性定义。

连接(续)

- 5）举例

[例5]

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

R

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

S

连接(续)

$$R \underset{C<E}{\bowtie} S$$

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

连接(续)

等值连接 $R \underset{R.B=S.B}{\bowtie} S$

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

连接(续)

自然连接 $R \bowtie S$

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

等值连接与自然连接的区别

- 1. 等值连接中不要求相等属性值的属性名相同，而自然连接要求相等属性值的属性名必须相同，即两关系只有在同名属性才能进行自然连接。
- 2. 等值连接不将重复属性去掉，而自然连接去掉重复属性，也可以说，自然连接是去掉重复列的等值连接。

除(续)

R

A	B	C
a_1	b_1	c_2
a_2	b_3	c_7
a_3	b_4	c_6
a_1	b_2	c_3
a_4	b_6	c_6
a_2	b_2	c_3
a_1	b_2	c_1

S

B	C	D
b_1	c_2	d_1
b_2	c_1	d_1
b_2	c_3	d_2

$R \div S$
A
a_1

课堂练习：

关系R

A	B
1	a
2	b
3	a
3	b
4	a

关系S

C
x
y

关系T

A
1
3

关系U

B	C
a	x
c	z

- **关系代数表达式R ×S ÷T-U的运算结果为：**

R ×S

A	B	C
1	a	x
1	a	y
2	b	x
2	b	y
3	a	x
3	a	y
3	b	x
3	b	y
4	a	x
4	a	y

R ×S ÷ T

B	C
a	x
a	y

R ×S ÷ T-U

B	C
a	y

课堂作业

- 已知： 学生关系S(SNO，SNAME，AGE，SDEPT)
  选课关系SC(SNO，CNO，GRADE)
  课程关系C(CNO，CNAME，CDEPT，TNAME)
- 试用关系代数式表示以下每个查询语句。
  - 查找计算机系的全体学生的学号、姓名和性别
  - 查找选修课程号为C2的学生学号与姓名
  - 查找选修课程名为“数据结构”的学生学号与姓名
  - 查找选修课程号为C2或C4的学生学号
  - 查找至少选修课程号为C2和C4的学生学号
  - 查找不学C2课程的学生性名与年龄
  - 查找选修全部课程的学生姓名
  - 查找所学课程包含计算机系所开设的全部课程的学生学号

答案

- 查找计算机系的全体学生的学号、姓名和性别

$$\Pi_{SNO,SNAME,SEX}(\sigma_{SDEPT = '计算机'}(S))$$

- 查找选修课程号为C2的学生学号与姓名

$$\Pi_{SNO,SNAME}(\sigma_{CNO = 'C2'}(S \bowtie SC))$$

- 查找选修课程名为“数据结构”的学生学号与姓名

$$\Pi_{SNO,SNAME}(\sigma_{CNAME = '数据结构'}(S \bowtie SC \bowtie C))$$

- 查找选修课程号为C2或C4的学生学号

$$\Pi_{SNO}(\sigma_{CNO = 'C2' \vee CNO = 'C4'}(SC))$$

答案

- 查找至少选修课程号为C2和C4的学生学号

$$\Pi_{SNO}(\sigma_{SC.SNO=T.SNO \wedge SC.CNO='C2' \wedge T.CNO='C4'}(SC \bowtie \rho_T(SC)))$$

- 查找不学C2课程的学生性名与年龄

$$\Pi_{SNAME,AGE}(S) - \Pi_{SNAME,AGE}(\sigma_{CNO='C2'}(S \bowtie SC))$$

- 查找选修全部课程的学生姓名

$$\Pi_{SNAME}(S \bowtie (\Pi_{SNO,CNO}(SC) \div \Pi_{CNO}(C)))$$

- 查找所学课程包含计算机系所开设的全部课程的学生学号

$$\Pi_{SNO,CNO}(SC) \div \Pi_{CNO}(\sigma_{CDEPT='计算机'}(C))$$

Example Queries

- 求未选修001号课程的学生号

方案1: $\Pi_{S\#}(S) - \Pi_{S\#}(\sigma_{C\# = 001}(SC))$

方案2: $\Pi_{S\#}(\sigma_{C\# \neq 001}(SC))$

哪一个正确？

S#	SN	AGE
s1	…	…
s2	…	…
s3	…	…

S#	C#	G
s1	001	90
s2	001	95
s1	002	96

S

S#	SNAME	AGE	SEX
1	李强	23	男
2	刘丽	22	女
5	张友	22	男

C

C#	CNAME	TEACHER
k1	C 语言	王华
k5	数据库原理	程军
k8	编译原理	程军

SC

S#	C#	GRADE
1	k1	83
2	k1	85
5	k1	92
2	k5	90
5	k5	84
5	k8	80

图 2.29　关系 S、C 和 SC

(1) 检索”程军”老师所授课程的课程号（C#）和课程名（CNAME）。

(2) 检索年龄大于21的男学生学号（S#）和姓名（SNAME）。

(3) 检索至少选修“程军”老师所授全部课程的学生姓名（SNAME）。

(4) 检索“李强”同学不学课程的课程号（C#）。

(5) 检索至少选修两门课程的学生学号（S#）。

(6) 检索全部学生都选修的课程的课程号（C#）和课程名（CNAME）。

(7) 检索选修课程包含”程军”老师所授课程之一的学生学号（S#）。

(8) 检索选修课程号为k1和k5的学生学号（S#）。

(9) 检索选修全部课程的学生姓名（SNAME）。

(10) 检索选修课程包含学号为2的学生所修课程的学生学号（S#）。

(11) 检索选修课程名为“C语言”的学生学号（S#）和姓名（SNAME）。

数据查询

- 语句格式

SELECT [ALL|DISTINCT] <目标列表达式>

[，<目标列表达式>] …

FROM <表名或视图名>[， <表名或视图名>] …

[WHERE <条件表达式>]

[GROUP BY <列名1> [HAVING <条件表达式>]]

[ORDER BY <列名2> [ASC|DESC]];

什么是视图

视图是一张虚拟表，它表示一张表的部分数据或多张表的综合数据，其结构和数据是建立在对表的查询基础上

视图中并不存放数据，而是存放在视图所引用的原始表（基表）中

同一张原始表，根据不同用户的不同需求，可以创建不同的视图

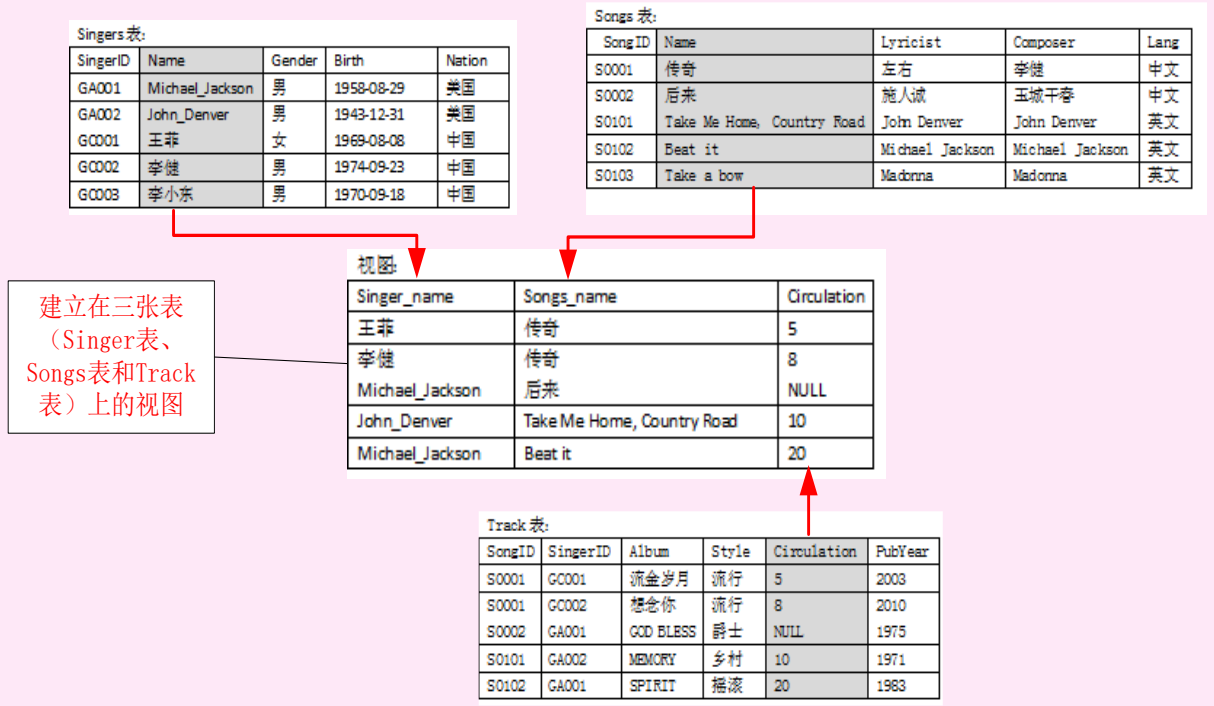
什么是视图

视图的用途

- 筛选表中的行
- 防止未经许可的用户访问敏感数据
- 降低数据库的复杂程度
- 将多个物理数据表抽象为一个逻辑数据表

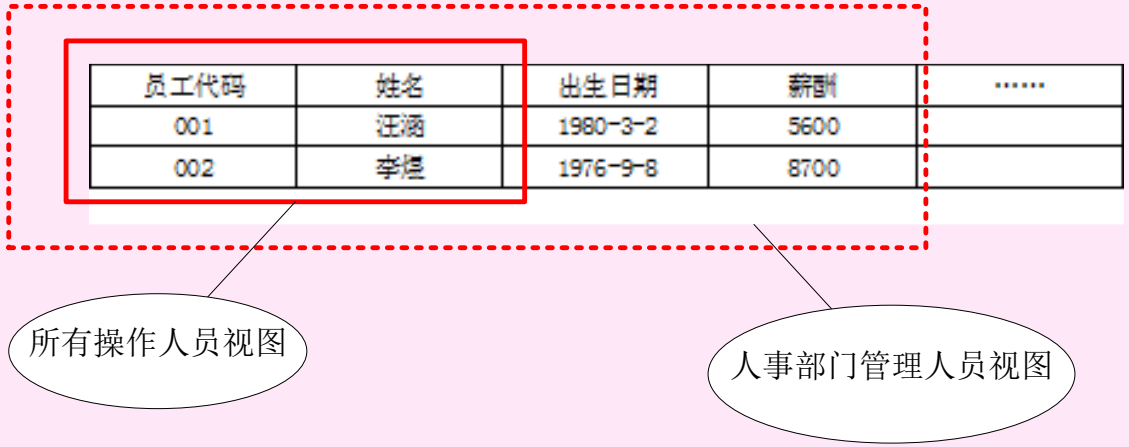
引入视图的原因

- 从业务数据角度来看
  - 由于数据库设计时考虑到数据异常等问题，同一种业务数据有可能被分散在不同的表中
  - 对这种业务数据的使用经常是同时使用的，要实现这样的查询，需要通过连接查询或嵌套查询来实现。



引入视图的原因（续）

- 从数据安全角度来看
  - 由于工作性质和需求不同，不同的操作人员只能查看表中的部分数据，不能查看表中的所有数据。



引入视图的原因（续）

- 从数据的应用角度来看
  - 一个报表中的数据往往来自于多个不同的表中。在设计报表时，需要明确地指定数据的来源途径和方式。通过视图机制，可以提高报表的设计效率。
- 视图是**RDBMS**提供给用户以多种角度来观察数据库中数据的重要机制。

一、建立视图

- 语句格式

  CREATE VIEW

  <视图名> [(<列名> [，<列名>]...)]

  AS <子查询>

  [WITH CHECK OPTION];

- **子查询不允许含有ORDER BY子句和DISTINCT短语**
- **RDBMS执行CREATE VIEW语句时只是把视图定义存入数据字典，并不执行其中的SELECT语句。**
- **在对视图查询时，按视图的定义从基本表中将数据查出。**

索引

- 实现数据快速查询的数据对象。
- 好处：
  - 索引是一个与表相关联的数据结构
  - 确保快速访问数据
  - 加快连接表的查询和完成排序和分组
  - 增强行的唯一性
  - 索引将占用用户数据库的空间
- 说明：
  - 索引总是在最常查询的列上创建
  - **Sql server 2005,** 在表上创建**primary key**或**unique**约束时，索引以与约束同样的名称被自动创建

数 据 控 制 (DCL)

❖通过对用户使用权限的限制而保证数据安全的重要措施。

■授权 (Grant)

■收回权限 (Revoke)

Grant

- 授权语句的一般格式为:

  GRANT　<权限>[,<权限>]…
  [ON <对象类型><对象名>]
  TO <用户>[,<用户>]…
  [WITH　GRANT　OPTION];

- 任选项WITH　GRANT　OPTION的作用是使获得某 种权力的用户可以把权力再授予别的用户.

- 对不同类型的操作对象可有不同的操作权力

对象类型	操作权力
表、视图、字段（TABLE）	SELECT，INSERT，UPDATE，DELETE，ALL PRIVILEGE
基本表（TABLE）	ALTER，INDEX
数据库（DATABASE）	CREATETAB
表空间（TABLESPACE）	USE
系统	CREATEDBC

例1. 把查询Student表的权限授予用户U1.

```
GRANT   SELECT
 ON  TABLE   Student
 TO  U1;
```

例2. 把对表Student和Course的全部操作权限授予U2和U3。

```
GRANT  ALL  PRIVILIGES
  ON  TABLE  Student, Course  TO  U2, U3;
```

例3. 把对表SC的查询权限授予所有用户.

```
GRANT   SELECT
 ON  TABLE   SC   TO  PUBLIC;
```

例4. 把修改学生学号和查询Student表的权限授给用户U4.

GRANT UPDATE(Sno), SELECT

ON TABLE Student TO U4;

例5 把对表SC的INSERT权限授予U5用户,并允许将此权限再授予其他用户.

GRANT INSERT

ON TABLE SC TO U5 WITH GRANT OPTION;

例6. DBA把在数据库S_C中建立表的权限授予用户U8.

GRANT CREATETAB ON DATABASE S_C TO U8;

SQL SERVER

- **GRANT { ALL [PRIVILEGES] } | *permission* [(*column* [,...n])] [,...n] [ON [*class* ::] *securable*] TO *principal* [,...n] [WITH GRANT OPTION] [AS *principal*]**

- ***class***
  - 指定将授予其权限的安全对象的类。
  - 需要范围限定符“::”。

SQL server中的用法

- 授予对表的 **SELECT** 权限
  - **GRANT SELECT ON OBJECT::Address TO RosaQdM;**

- 授予对存储过程的 **EXECUTE** 权限
  - **GRANT EXECUTE ON OBJECT::uspUpdateEmployeeHireInfo TO Recruiting11;**

回收权限

- 授予的权力可以用REVOKE语句收回
- 格式为：

```
REVOKE    〈权限〉[, 〈权限〉]…
 [ON  〈对象类型〉〈对象名〉]
 FROM  〈用户〉[,〈用户〉]…;
```

例7. 把用户U4修改学生学号的权力收回.

```
REVOKE UPDATE(Sno) ON TABLE Studeny FROM U4;
```

例8. 收回所有用户对表SC的查询权限.

```
REVOKE SELECT ON TABLE Student FROM PUBLIC;
```

例9. 把用户U5对表SC的INSERT权限收回.

```
REVOKE INSERT ON TABLE SC FROM U5;
```

函数依赖的定义及性质

- 在关系模式SCD中，SNO与SN、AGE、DEPT之间都有一种依赖关系。
- 由于一个SNO只对应一个学生，而一个学生只能属于一个系，所以当SNO的值确定之后，SN，AGE，DEPT的值也随之被唯一的确定了。
- 这类似于变量之间的单值函数关系。设单值函数Y=F(X)，自变量X的值可以决定一个唯一的函数值Y。
- 在这里，我们说SNO决定函数（SN，AGE，DEPT），或者说（SN，AGE，DEPT）函数依赖于SNO。

函数依赖的形式化定义

定义5　设关系模式R(U，F)，U是属性全集，F是U上的函数依赖集，X和Y是U的子集，如果对于R(U)的任意一个可能的关系r，对于X的每一个具体值，Y都有唯一的具体值与之对应，则称X决定函数Y，或Y函数依赖于X，记作X→Y。我们称X为决定因素，Y为依赖因素。当Y不函数依赖于X时，记作：X↛Y。

若X → Y,且Y　⊆X,则称X→Y是平凡的函数依赖

若X → Y,但Y　⊄X,则称X→Y是非平凡的函数依赖，若不特别声明，总是讨论非平凡的函数依赖

当X→Y且Y→X时，则记作：X↔Y

有关函数依赖的说明：

1. 函数依赖是语义范畴的概念。
   - 只能根据语义来确定一个函数依赖，而不能按照其形式化定义来证明一个函数依赖是否成立。
   - 例如，对于关系模式S，当学生不存在重名的情况下，可以得到：
     - SN→AGE
     - SN→DEPT
   - 这种函数依赖关系，必须是在没有重名的学生条件下才成立的，否则就不存在函数依赖了。
   - 所以函数依赖反映了一种语义完整性约束。

有关函数依赖的说明：

2. 函数依赖与属性之间的联系类型有关。

- （1）在一个关系模式中，如果属性X与Y有1:1联系时，则存在函数依赖X→Y，Y→X，即X←→Y。
- 例如，当学生无重名时，SNO ↔ SN。
- （2）如果属性X与Y有m:1的联系时，则只存在函数依赖X→Y。
- 例如，SNO与AGE，DEPT之间均为m:1联系，所以有SNO→AGE，SNO→DEPT。
- （3）如果属性X与Y有m: n的联系时，则X与Y之间不存在任何函数依赖关系。
- 例如，一个学生可以选修多门课程，一门课程又可以为多个学生选修，所以SNO与CNO之间不存在函数依赖关系
- 可从属性间的联系类型来分析属性间的函数依赖

关系的键（码）

定义 5.1　设K为关系模式$R(U,F)$中的属性或属性组合。若K满足以下条件，则称K为R的一个**候选键（Candidate Key）**：

（1）$K \rightarrow U$；

（2）K的任意真子集都不能函数决定R的其他属性，即K满足最小化条件。

若关系模式R有多个候选码，则选定其中的一个做为**主键（Primary key）**。

- **例5- 3**：在SCD(学号，姓名，年龄，系名，系主任，课程号，成绩)关系中，候选键是属性集{学号，课程号}

- 首先，证明{学号，课程号}函数决定其他的属性
- 其次，证明{学号，课程号}的任何真子集都不能函数决定所有其他的属性。

- 例5- 4：确定关系模式Project（项目编号，项目名称，员工编号，员工姓名，工资级别，工资）的候选键。

- 函数依赖：项目编号→ 项目名称，员工编号→ 员工姓名，员工编号→ 工资级别，工资级别→ 工资

- Project关系的候选键为属性集{项目编号，员工编号}。

- 主属性与非主属性
  - 包含在任何一个候选码中的属性，称为主属性（Prime attribute）
  - 不包含在任何码中的属性称为非主属性（Nonprime attribute）或非码属性（Non-key attribute）
- ALL KEY
  - 整个属性组是码，称为全码（All-key）

- 不重名的条件下， SCD(学号，姓名，年龄，系名，系主任，课程号，成绩)关系的主属性？

- 可以有两个候选键：{学号，课程号}和{姓名，课程号}，那么，学号，课程号，姓名都是主属性，其余属性是非主属性。

数据依赖的公理系统

- W. W. Armstrong于1974年提出了一套推理规则。使用这套规则，可以由已有的函数依赖推导出新的函数依赖。
- Armstrong公理
  - X，Y，Z是属性集，
    - 自反律。若$Y \subseteq X$， 则$X \rightarrow Y$。
    - 增广律。若$X \rightarrow Y$，则$XZ \rightarrow YZ$。
    - 传递律。若$X \rightarrow Y$， $Y \rightarrow Z$，则$X \rightarrow Z$ 。

数据依赖的公理系统

由Armstrong公理导出的推理规则

合并律：若$X \to Y$，$X \to Z$，则$X \to YZ$。

分解律：若$X \to YZ$，则$X \to Y$，$X \to Z$。

伪传递律：若$X \to Y$，$WY \to Z$，则$XW \to Z$。

求闭包的算法

算法 5.1

输入：X，F

输出：X^+

步骤：

$X^+ := X$；

while (X^+ 发生变化）do

for each 函数依赖 $A \rightarrow B$ in F do

begin

if $A \subseteq X^+$ then $X^+ := X^+ \bigcup B$

end

- 结论:
  判定函数依赖X→Y是否能由F导出的问题，
  可转化为求X^+并判定Y是否是X^+子集的问题。即求闭包问题可转化为求属性集问题。

求关键字问题可转化为求属性集闭包问题，

因为$X \rightarrow X^+$，若$U=X^+$，则$X \rightarrow U$，即X为关键字。

例：求关系模式R=(ABCD,{A→B,B→C})的关键字

验证：(AD)+=ABCD

- Consider a relational schema ABCDEFGHIJ, which contains the following FDs:
- AB$\rightarrow$C, D$\rightarrow$E, AE$\rightarrow$G, GD$\rightarrow$H, IF$\rightarrow$J.
-
- Check whether or not the functional dependencies entail
  - ABD$\rightarrow$GH
  - ABD$\rightarrow$HJ
  - ABC$\rightarrow$G
  - GD$\rightarrow$HE

关系模式的存储异常问题

- 分解后的关系模式是一个好的关系数据库模式。
- 一个好的关系模式应该具备以下四个条件：
  - 1. 尽可能少的数据冗余。
  - 2. 没有插入异常。
  - 3. 没有删除异常。
  - 4. 没有更新异常。

关系模式的存储异常问题

- 注意：一个好的关系模式并不是在任何情况下都是最优的，
  - 比如查询某个学生选修课程名及所在系的系主任时，要通过连接，而连接所需要的系统开销非常大，因此要以实际设计的目标出发进行设计
- 如何按照一定的规范设计关系模式，将结构复杂的关系分解成结构简单的关系，降低或消除数据库中冗余数据的过程，这就是关系的规范化。

函数依赖类型1

- 定义6.2 在R(U)中，如果X→Y，并且对于X的任何一个真子集X’，都有X’$\not\rightarrow$Y，则称Y对X完全函数依赖，记作 $X \xrightarrow{f} Y$。

  若X→Y，但Y不完全函数依赖于X，则称Y对X部分函数依赖，记作$X \xrightarrow{p} Y$。

- 在一个关系模式中，当存在非主属性对候选键的部分依赖时，就会产生数据冗余和更新异常。
- 若非主属性对候选键完全函数依赖，不会出现以上问题。

- 由上可知：
  - 只有当决定因素是组合属性时，讨论部分函数依赖才有意义
  - 当决定因素是单属性时，只能是完全函数依赖。
    - 例如，在关系模式S（SNO，SN，AGE，DEPT），决定因素为单属性SNO，有SNO→（SN，AGE，DEPT），不存在部分函数依赖。

函数依赖类型2

- 传递函数依赖：在R(U)中，如果X→Y，Y → Z，且Y$\ddot{E}$ X，Y ↛ X，则称Z对X传递函数依赖。

- 如果Y→X，则X ←→Y，这时称Z对X直接函数依赖，而不是传递函数依赖。
- 例如，在关系模式SCD中，SNO→DEPT，但DEPT↛SNO，而DEPT→MN，则有SNO $\xrightarrow{t}$ MN。
- 当学生不存在重名的情况下，有SNO→SN，SN→SNO，SNO↔SN，SNO→DEPT,则SN→DEPT，这时DEPT对SN是直接函数依赖，而不是传递函数依赖。

函数依赖类型

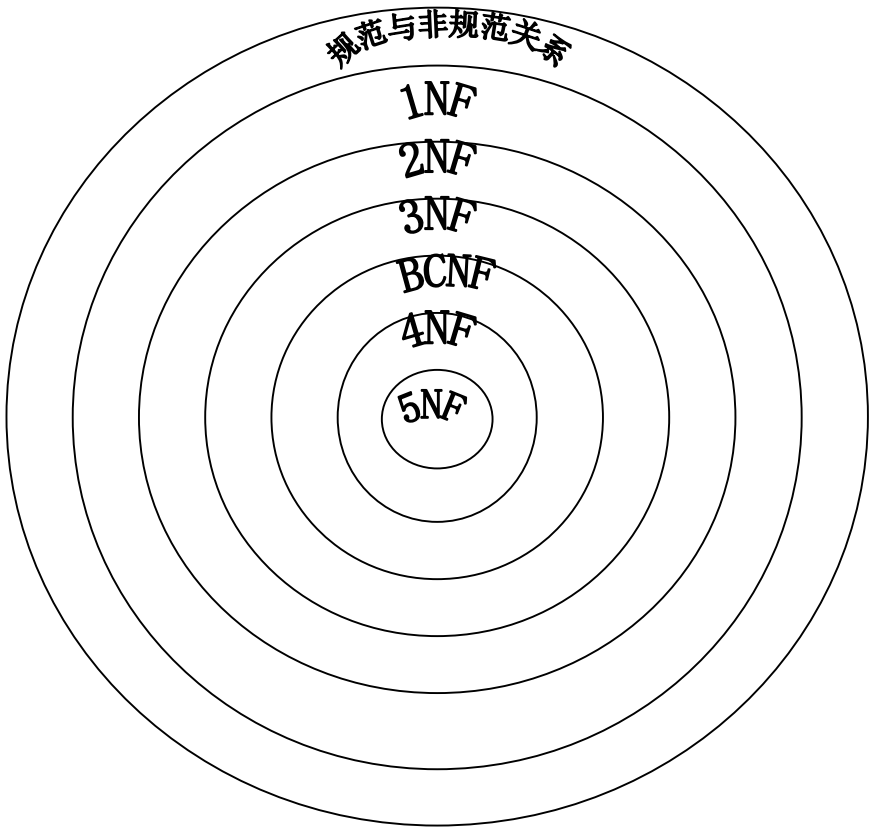
- 综上所述，函数依赖分为完全函数依赖、部分函数依赖和传递函数依赖三类，它们是规范化理论的依据和规范化程度的准则
- 从函数依赖的角度看，关系模式中存在各属性（包括非主属性和主属性）对候选键的部分依赖和传递依赖是产生数据冗余和更新异常的根源。

范式

- 规范化的基本思想是消除关系模式中的数据冗余，消除数据依赖中的不合适的部分，解决数据插入、删除时发生异常现象。
- 这就要求关系数据库设计出来的关系模式要满足一定的条件。
- 关系数据库的规范化过程中为不同程度的规范化要求设立的不同标准称为范式（Normal Form）。
- 由于规范化的程度不同，就产生了不同的范式。
- 满足最基本规范化要求的关系模式叫第一范式，
- 在第一范式中进一步满足一些要求为第二范式，
- 以此类推就产生了第三范式等概念。

范式之间的关系

- 各个范式之间的联系可以表示为：
  - 5NF⊂4NF ⊂ BCNF ⊂ 3NF ⊂ 2NF ⊂ 1NF



- 某一关系模式R为第n范式，可简记为R∈nNF。
- 一个低一级范式的关系模式，通过模式分解可以转换为若干个高一级范式的关系模式的集合，这种过程就叫规范化

第一范式

- 第一范式（First Normal Form）是最基本的规范形式，即关系中每个属性都是不可再分的简单项。
- 定义8　如果关系模式R，其所有的属性均为简单属性，即每个属性是不可再分的，则称R属于第一范式，简称1NF，记作R∈1NF。
- 在非规范化的关系中去掉组合项就能化成规范化的关系。每个规范化的关系都属于1NF，这也是它之所以称为“第一”的原因。

- 第一范式是对关系模式的最起码的要求。不满足第一范式的数据库模式不能称为关系数据库
- 但是满足第一范式的关系模式并不一定是一个好的关系模式

第二范式

- 定义9　如果关系模式R∈1NF，且每个非主属性都完全函数依赖于R的每个关系键，则称R属于第二范式（Second Normal Form），简称2NF，记作R∈2NF。

- 在关系模式SCD中，SNO，CNO为主属性，AGE，DEPT，MN，MN，SCORE均为非主属性，经上述分析，存在非主属性对关系键的部分函数依赖，所以SCD∉2NF。

第二范式

- 结论：
  - 1．从1NF关系中消除非主属性对关系键的部分函数依赖，则可得到2NF关系。
  - 2．如果R的关系键为单属性，或R的全体属性均为主属性，则R∈2NF。

第三范式

定义10　如果关系模式R∈2NF，且每个非主属性都不传递依赖于R的每个关系键，则称R属于第三范式（Third Normal Form），简称3NF，记作R∈3NF。

- 第三范式具有如下性质：

1. 如果R∈3NF，则R也是2NF。
2. 如果R∈2NF，则R不一定是3NF。
   - 例如，我们前面由关系模式SCD分解而得到的SD和SC都为2NF，其中，SC∈3NF，但在SD中存在着非主属性MN对主键SNO传递依赖，SD ∈ 3NF。对于SD，应该进一步进行分解，使其转换成3NF。

3NF规范化

- 3NF规范化是指把2NF关系模式通过投影分解转换成3NF关系模式的集合。
- 和2NF的规范化时遵循的原则相同，即“一事一地”，让一个关系只描述一个实体或者实体间的联系。
- 3NF规范化是指把2NF关系模式通过投影分解转换成3NF关系模式的集合，因此要消除非主属性对候选键的传递依赖。分解的方法很简单，即以构成传递链的两个基本依赖为基础形成两个新的模式。

- 将SD(SNO, SN, AGE, DEPT, MN)规范到3NF。
  - 分析SD的属性组成，可以判断，关系SD实际上描述了两个实体：
  - 一个为学生实体，属性有SNO，SN，AGE，DEPT；
  - 另一个是系的实体，其属性DEPT和MN。

BC范式

- 3NF只限制了非主属性对键的依赖关系，而没有限制主属性对键的依赖关系。
- 如果发生了这种依赖，仍有可能存在数据冗余、插入异常、删除异常和修改异常。
- 需对3NF进一步规范化，消除主属性对键的依赖关系，为了解决这种问题，Boyce与Codd共同提出了一个新范式的定义，这就是Boyce-Codd范式，通常简称BCNF或BC范式。它弥补了3NF的不足

BC范式的定义和性质

定义11　如果关系模式R∈1NF，且所有的函数依赖X→Y（$Y \subseteq X$），决定因素X都包含了R的一个候选键，则称R属于BC范式（Boyce-Codd Normal Form），记作R∈BCNF。

BCNF具有如下性质：

- 1．满足BCNF的关系将消除任何属性（主属性或非主属性）对键的部分函数依赖和传递函数依赖。也就是说，如果R∈BCNF，则R也是3NF。

3NF与BCNF的关系

- R ∈BCNF ⇄ R ∈3NF（充分；不必要）

- 如果R∈3NF，且R只有一个候选码

  R ∈BCNF ⇄ R ∈3NF（充分；必要）

BC范式的定义和性质

2．如果R∈3NF，则R不一定是BCNF。

- 现举例说明。设关系模式SNC（SNO，SN，CNO，SCORE），其中SNO代表学号，SN代表学生姓名并假设没有重名，CNO代表课程号，SCORE代表成绩。可以判定，SNC有两个候选键（SNO，CNO）和（SN，CNO），其函数依赖如下：
  - SNO↔SN
  - （SNO，CNO）→SCORE
  - （SN，CNO）→SCORE。
- 但是，存在着主属性对键的部分函数依赖：
- （SNO，CNO）$\xrightarrow{P}$ SN，（SN，CNO）$\xrightarrow{P}$ SNO，所以SNC不是BCNF。

- 这种主属性对键的部分函数依赖关系，造成了关系SNC中存在着较大的数据冗余，学生姓名的存储次数等于该生所选的课程数。从而会引起修改异常。
- 解决这一问题的办法仍然是通过投影分解进一步提高SNC的范式等级，将SNC规范到BCNF。

要使关系模式属于 BC 范式，既要消除非主属性对于候选键的部分依赖和传递依赖，又要消除主属性对候选键的部分依赖和传递依赖。

由定义 5.13 可知，既然关系模式 R 不属于 BC 范式，那么至少能找到一个违背 BC 范式的函数依赖（这种依赖被称为违例）。那么分解就以违例为基础，方法如下：

① 把违例涉及到的所有属性（即该违例的决定因素和可以加到该违例右边的所有属性）组合成一个新的模式；

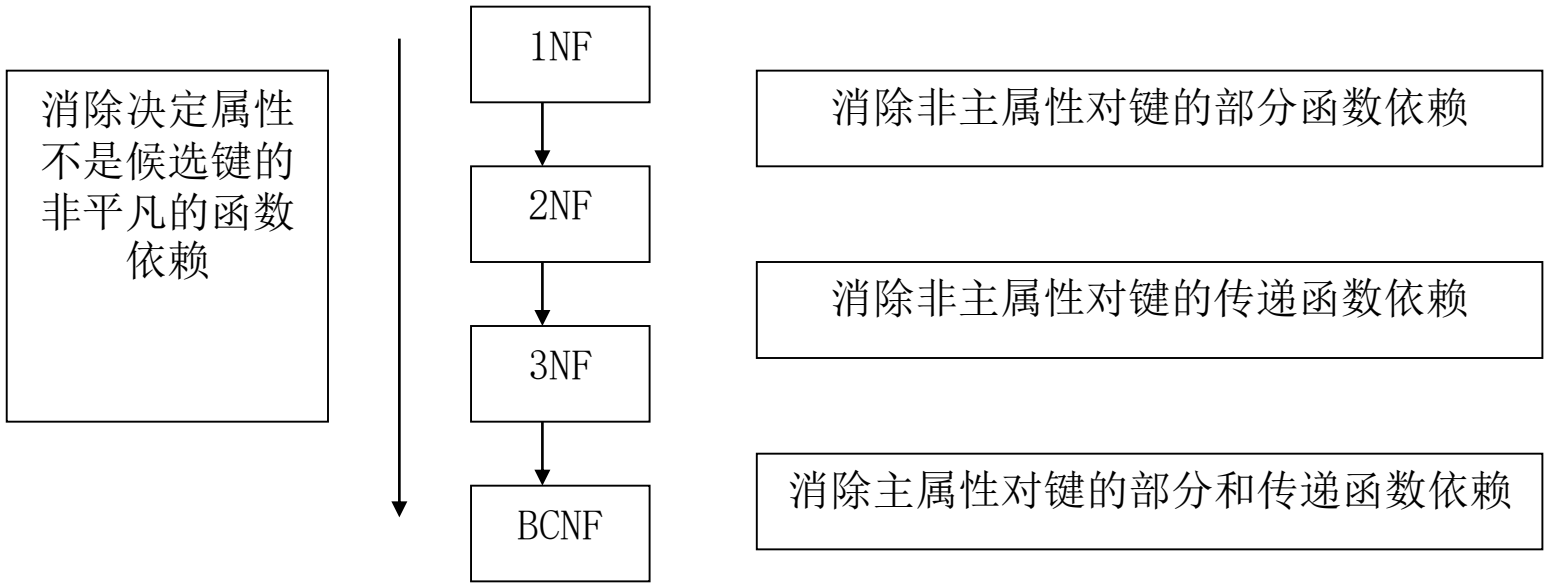
② 从属性全集中去掉违例的右边属性，组成另一个新模式。

以 SNC 为例说明规范化到 BC 范式的过程。

在 SNC 中，违例为函数依赖：SNO↔SN，因此，将 SNO 和 SN 组合成一个新的模式 S1(SNO，SN)，另一个模式为 S2(SNO，CNO，SCORE)。

分解后的关系模式 S1，有两个候选键 SNO 和 SN；关系模式 S2，主键为（SNO，CNO）。在这两个关系中，无论主属性还是非主属性都不存在对键的部分依赖和传递依赖，因此，S1∈BCNF，S2∈BCNF。

- 3NF和BCNF都是以函数依赖为基础来衡量关系模式规范化的程度。
- 一个关系模式如果属于BCNF，那么在函数依赖范畴内，已经实现了彻底的分离；
- 3NF由于可能存在主属性对键的部分依赖和传递依赖，因此关系模式的分解仍不够彻底


消除决定属性不是候选键的非平凡的函数依赖
1NF
2NF
3NF
BCNF
消除非主属性对键的部分函数依赖
消除非主属性对键的传递函数依赖
消除主属性对键的部分和传递函数依赖

- For the following relation schema and sets of FD's:
- R = (A, B, C, D, E). F ={$A \rightarrow BC$, $CD \rightarrow E$, $B \rightarrow D$, $E \rightarrow A$ }.

- Consider the relation R(A,B,C,D,E) with the following functional dependencies:
  (A, B)->E, (C, D)->E, A->C, C->A.
  Is R in BCNF?

ER模型

- 数据库设计中广泛使用的概念模型当属E-R数据模型
- E-R数据模型（Entity-Relationship data model），即实体联系数据模型，是P.P.S. Chen于1976年提出的一种语义数据模型。

ER模型介绍

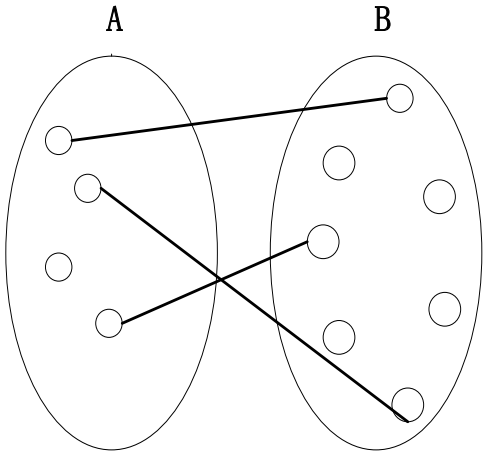
- 设计E-R数据模型的目标是有效地和自然地模拟现实世界
- E-R数据模型只应包含那些对描述现实世界有普遍意义的抽象概念
- 三要素：
  - 实体
  - 联系
  - 属性

二元联系

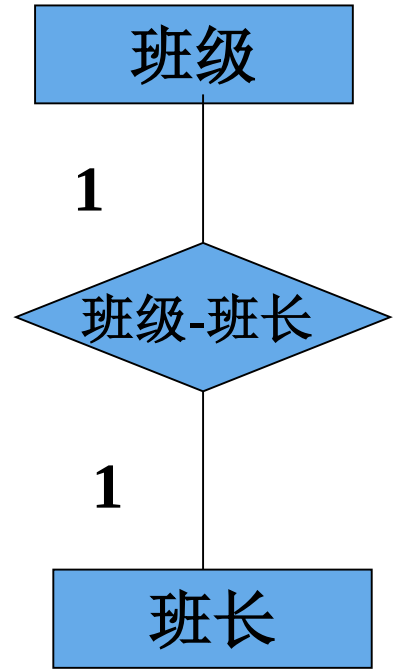
- 只有两个实体集参与的联系称为二元联系，它是现实世界中大量存在的联系
- 二元联系可进一步细分为以下三种联系：
  - 一对一（1：1）联系
  - 一对多（1：n）的联系
  - 多对多 (m：n)联系

一对一联系

- 如果对于实体集A中的每一个实体，实体集B中至多有一个实体与之联系，反之亦然，则称实体集A与实体集B 具有一对一联系。记为1:1。



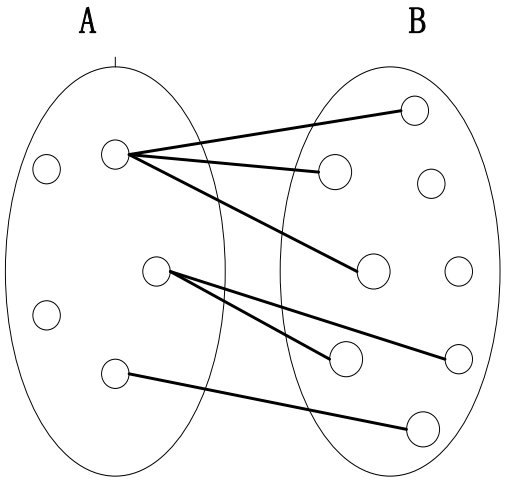
(a)一对一联系



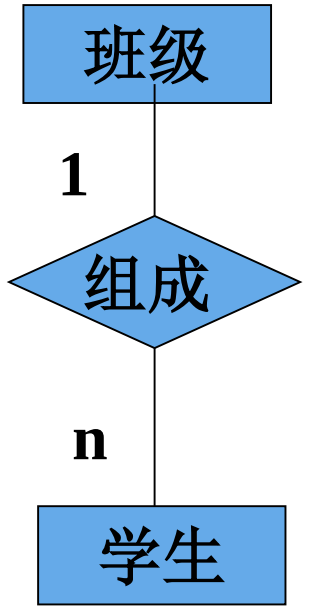
1:1联系

一对多（1：n）的联系

- 设有两个实体集A和B，若A中每个实体与B中任意个实体（包括零个）发生关联，但B中每个实体至多和A中一个实体有联系、则称A和B是一对多的联系，记为1：n。



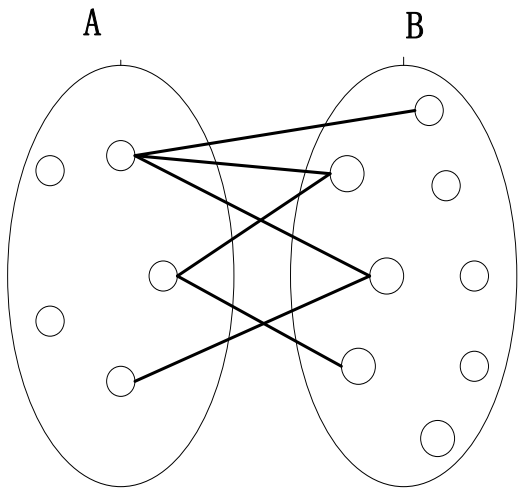
(b)一对多联系



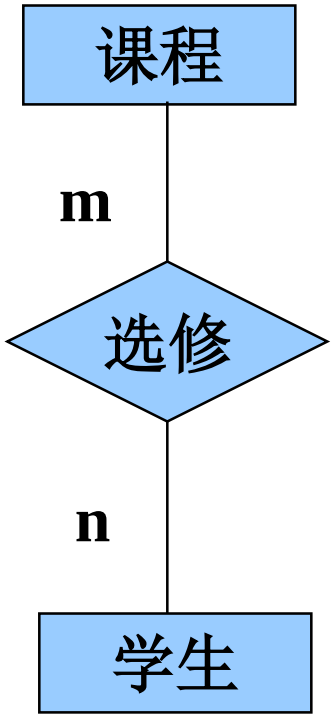
1:n联系

多对多 (m：n)联系

- 若两个实体集A和B，每个实体集中的每一个实体都和另一个实体集中任意多个（包括0个）实体有联系，则称A和B是多对多的联系，记为m：n。



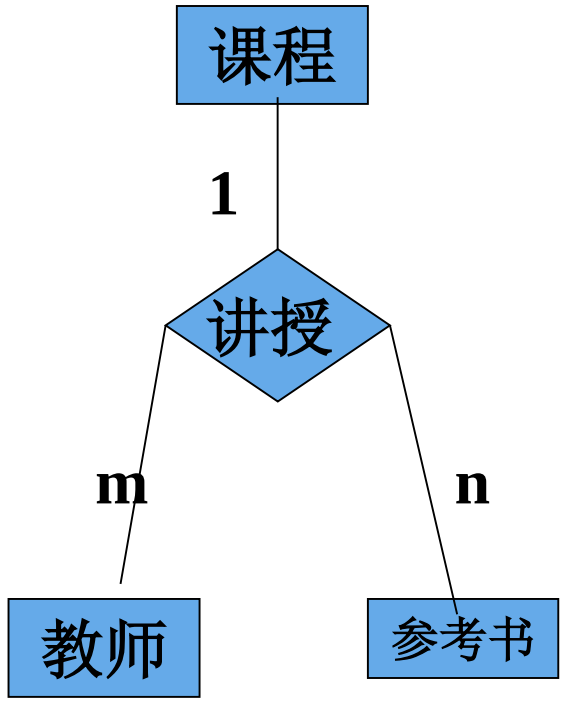
(c)多对多联系



m:n联系

多个实体型间的联系(续)

- 实例
  - 课程、教师与参考书三个实体型
  - 如果一门课程可以有若干个教师讲授，使用若干本参考书，每一个教师只讲授一门课程，每一本参考书只供一门课程使用课程与教师、参考书之间的联系是一对多的



两个以上实体型间1:n联系

同一实体型内部的联系

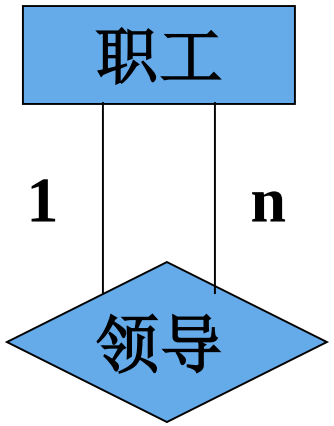
- 实例

  职工实体型内部具有领导与被领导的联系

  某一职工（干部）“领导”若干名职工;

  一个职工仅被另外一个职工直接领导

  这是一对多的联系



单个实体型内部1:n联系

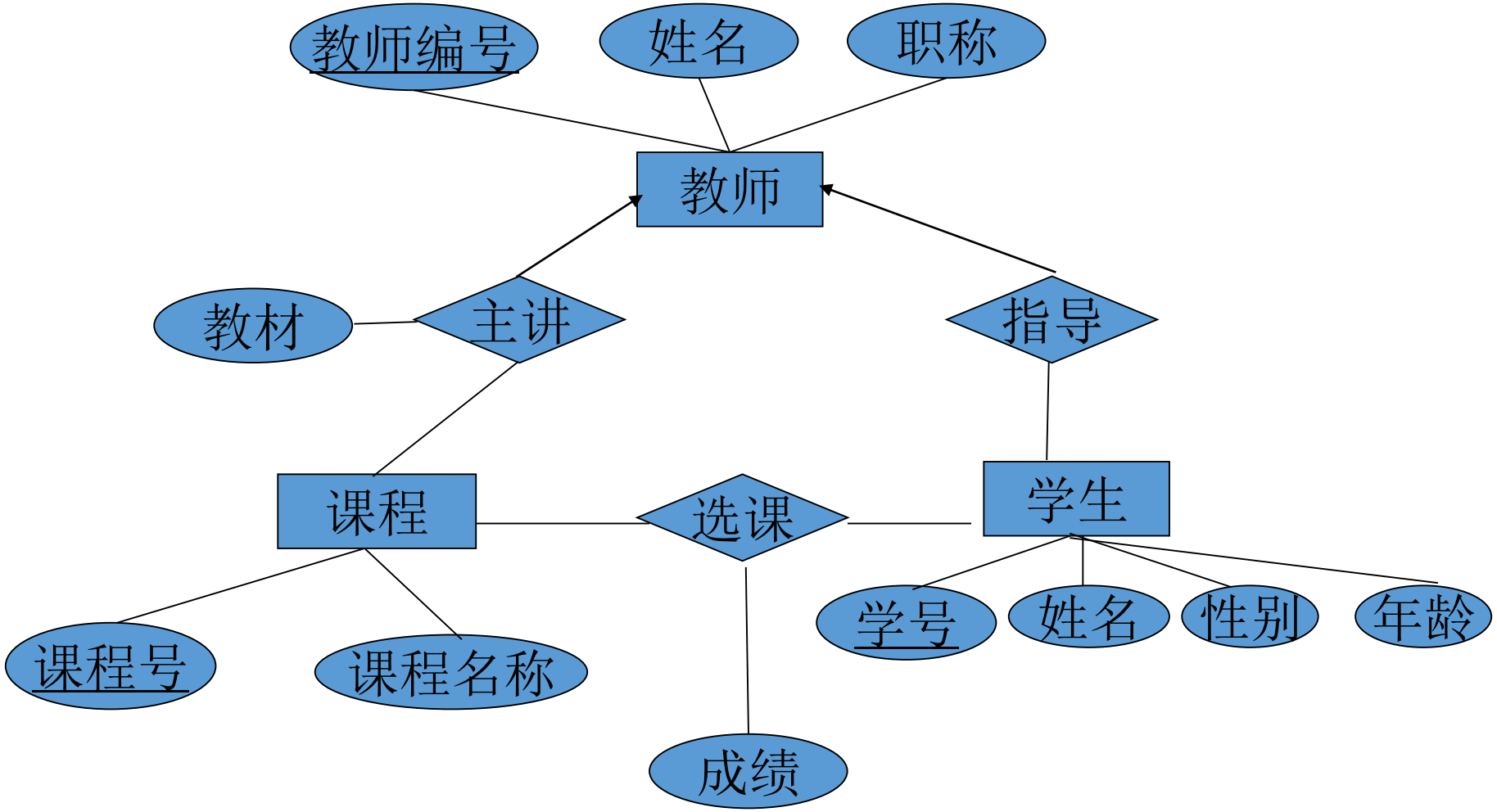
从ER模型到关系模型的转换

- 属于逻辑设计
- 基本原则
  - ER图中的每个实体转换为一个关系模式
    - 实体的属性即关系模式的属性
    - 实体标识符即关系模式的键
    - 联系的转换根据不同情况做不同的处理
      - 联系是1:1的，在任意一个关系模式的属性中加入另一个关系模式的键和联系的属性；
      - 联系是1:n的，在N端实体类型转换的关系模式中加入1端的关系模式的键和联系类型的属性；
      - 联系是m:n的，将联系类型也转换为关系模型，其属性为两端实体类型的键加上联系类型，键为两端实体键的组合。

Examples of E-R Diagram—大学教学管理数据库

- 实体集：
  - “学生”实体集：属性有学号、姓名、性别、年龄；
  - “教师”实体集：属性有教师编号、姓名、职称等；
  - “课程”实体集，属性有课程号、课程名称。
- 联系描述如下：
  - 教师与课程之间有“主讲”联系，每位教师可主讲若干门课程，但每门课程只有一位主讲教师，教师主讲课程将选用某本教材；
  - 教师与学生之间有“指导”联系，每位教师可指导若干学生，但每个学生只有一位指导教师；
  - 学生与课程之间有“选课”联系，每个学生可选修若干课程，每门课程可由若干学生选修，学生选修课程有个成绩。

Examples of E-R Diagram—大学教学管理数据库



E-R图实例：某工厂物资管理概念模型

用E-R图表示某个工厂物资管理的概念模型

- 实体
  - 仓库： 仓库号、面积、电话号码
  - 零件 ：零件号、名称、规格、单价、描述
  - 供应商：供应商号、姓名、地址、电话号码、帐号
  - 项目：项目号、预算、开工日期
  - 职工：职工号、姓名、年龄、职称

E-R图实例：某工厂物资管理概念模型

- 实体之间的联系如下：

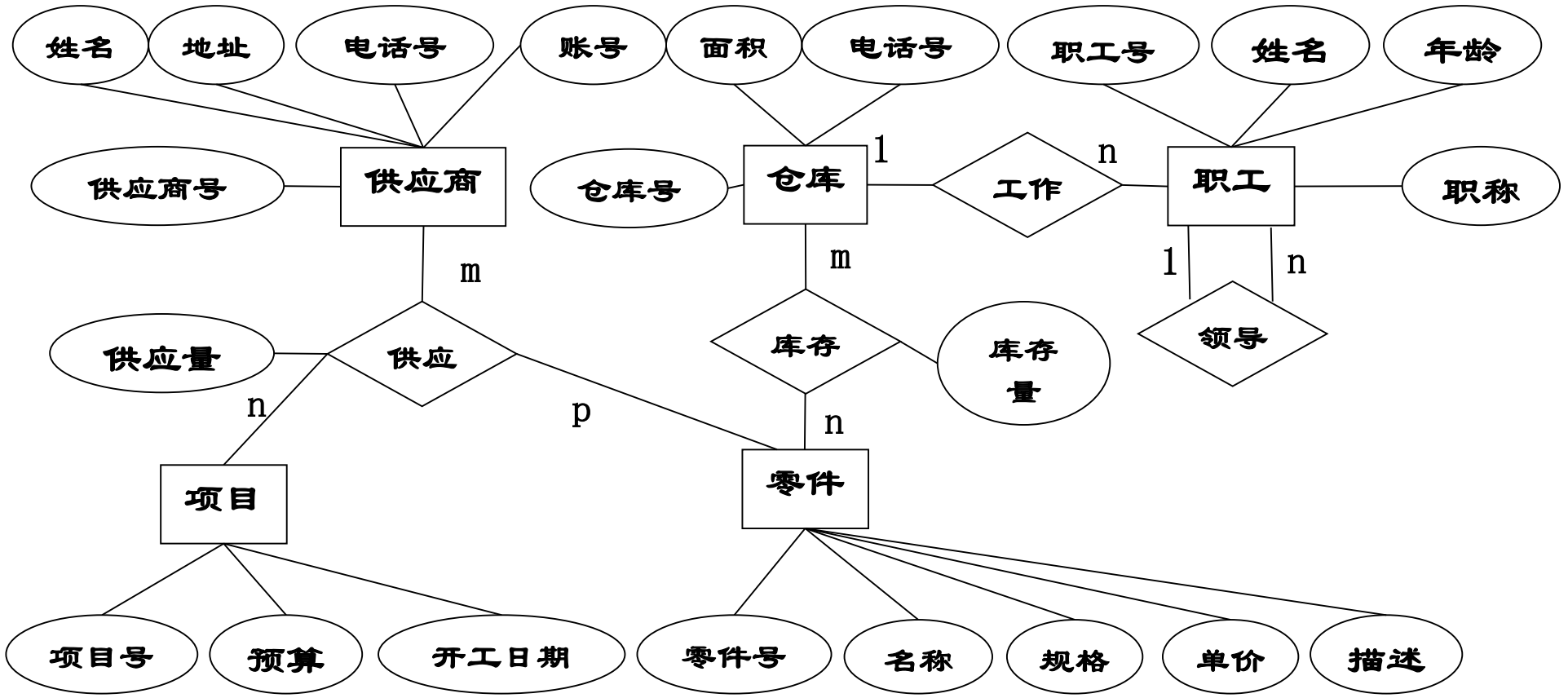
(1)一个仓库可以存放多种零件，一种零件可以存放在多个仓库中。用库存量来表示某种零件在某个仓库中的数量。

(2)一个仓库有多个职工当仓库保管员，一个职工只能在一个仓库工作。职工实体型中具有一对多的联系

(3)职工之间具有领导-被领导关系。即仓库主任领导若干保管员。

(4)供应商、项目和零件三者之间具有多对多的联系

E-R图实例：某工厂物资管理概念模型



在简单的教务管理系统中，包括四个实体，分别为：

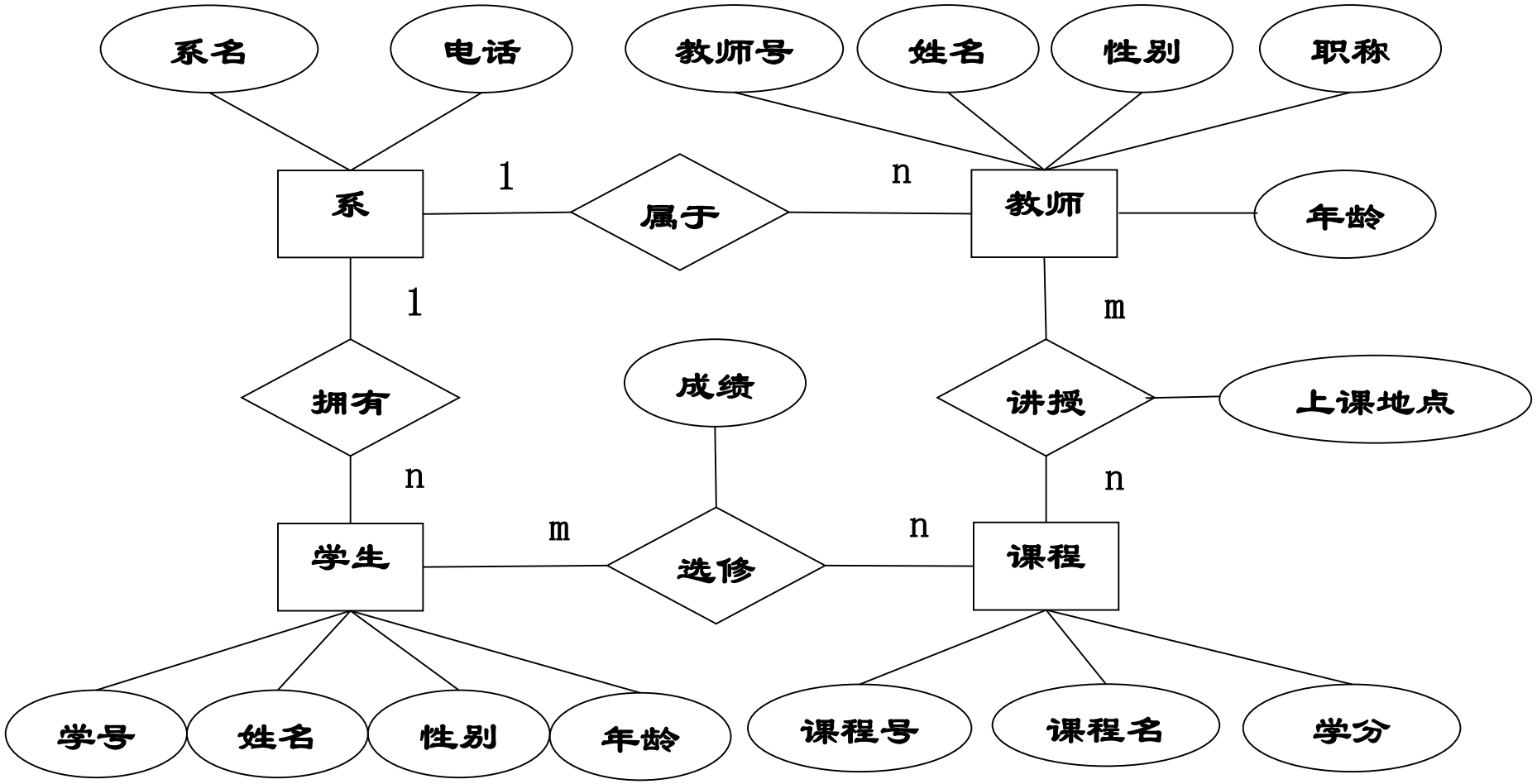
- 系：系名，电话
- 教师：教师号，姓名，性别，职称，年龄
- 学生：学号，姓名，性别，年龄
- 课程：课程号，课程名，学分

且存在如下语义约束：

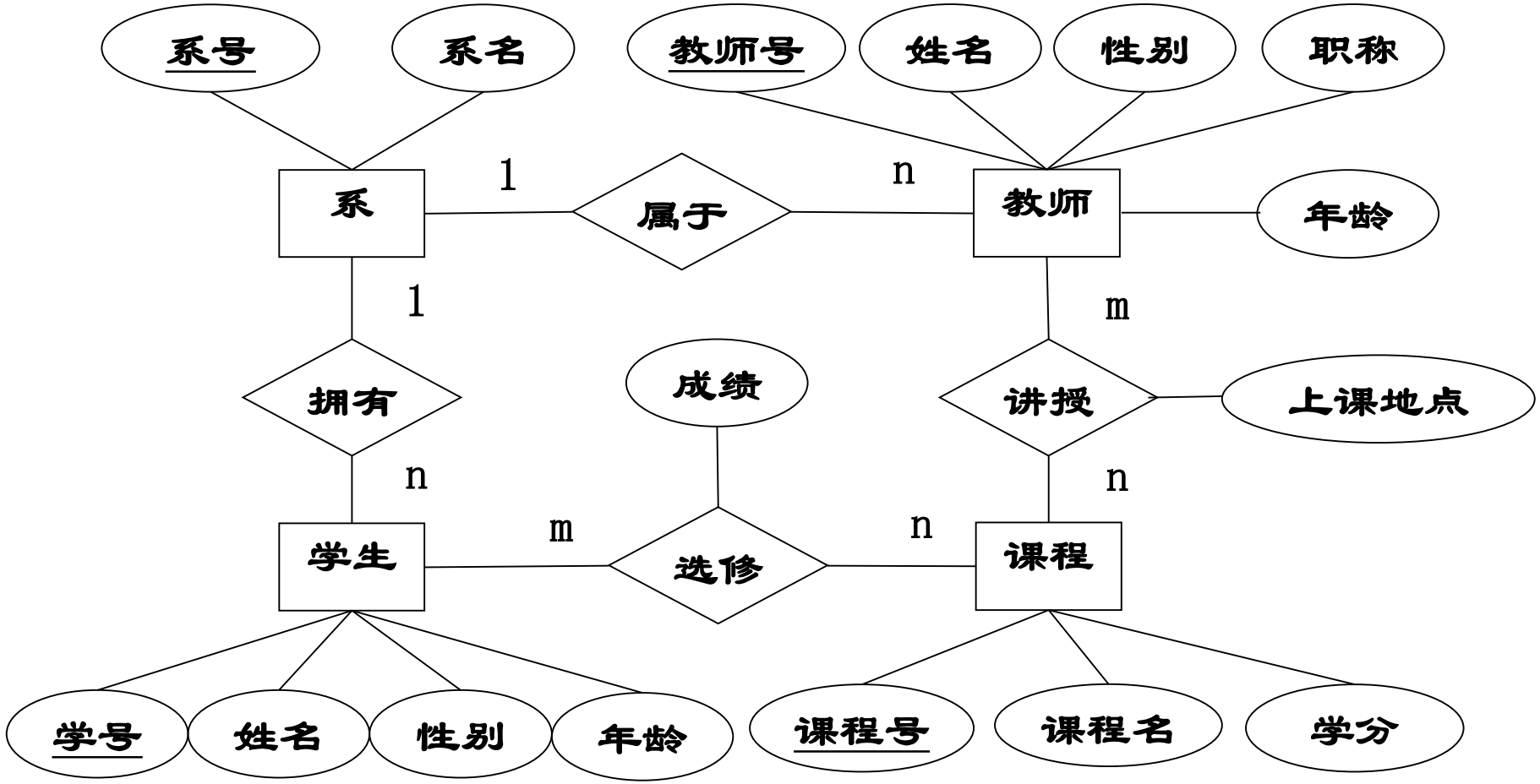
- 一个系可拥有多个教师，一个教师只能属于一个系。
- 一个系可拥有多个学生，一个学生只能属于一个系。
- 一个学生可选修多门课程，一门课程可为多个学生选修，每一个学生选修每门课程都有一个成绩。
- 一个教师可讲授多门课程，一门课程可为多个教师讲授。

画出E-R图。

简单教务管理系统的E－R图



将 ER 图映射到表


系号
系名
教师号
姓名
性别
职称
系
1
属于
n
教师
年龄
1
m
拥有
成绩
讲授
上课地点
n
n
学生
m
选修
n
课程
学号
姓名
性别
年龄
课程号
课程名
学分

order no	date	customer no	customer name	item	product no	product name	unit	unit price	amount	price
0610248	2006-5-30	VICRP	Victor Corp.	1	10001042	rice	3 Kg/bag	150	10	1500
0610248	2006-5-30	VICRP	Victor Corp.	2	10001072	coke	24 cans/box	480	5	2400
0610249	2006-6-8	DONDI	Dondi Corp.	1	10001014	milk	24 bottles	600	9	5400
0610249	2006-6-8	DONDI	Dondi Corp.	2	10001051	corn chips	24 bags/box	720	5	3600
0610249	2006-6-15	JENRE	Jenren Corp.	1	10001002	beer	24 cans/box	480	10	4800

where the keys are underlined.

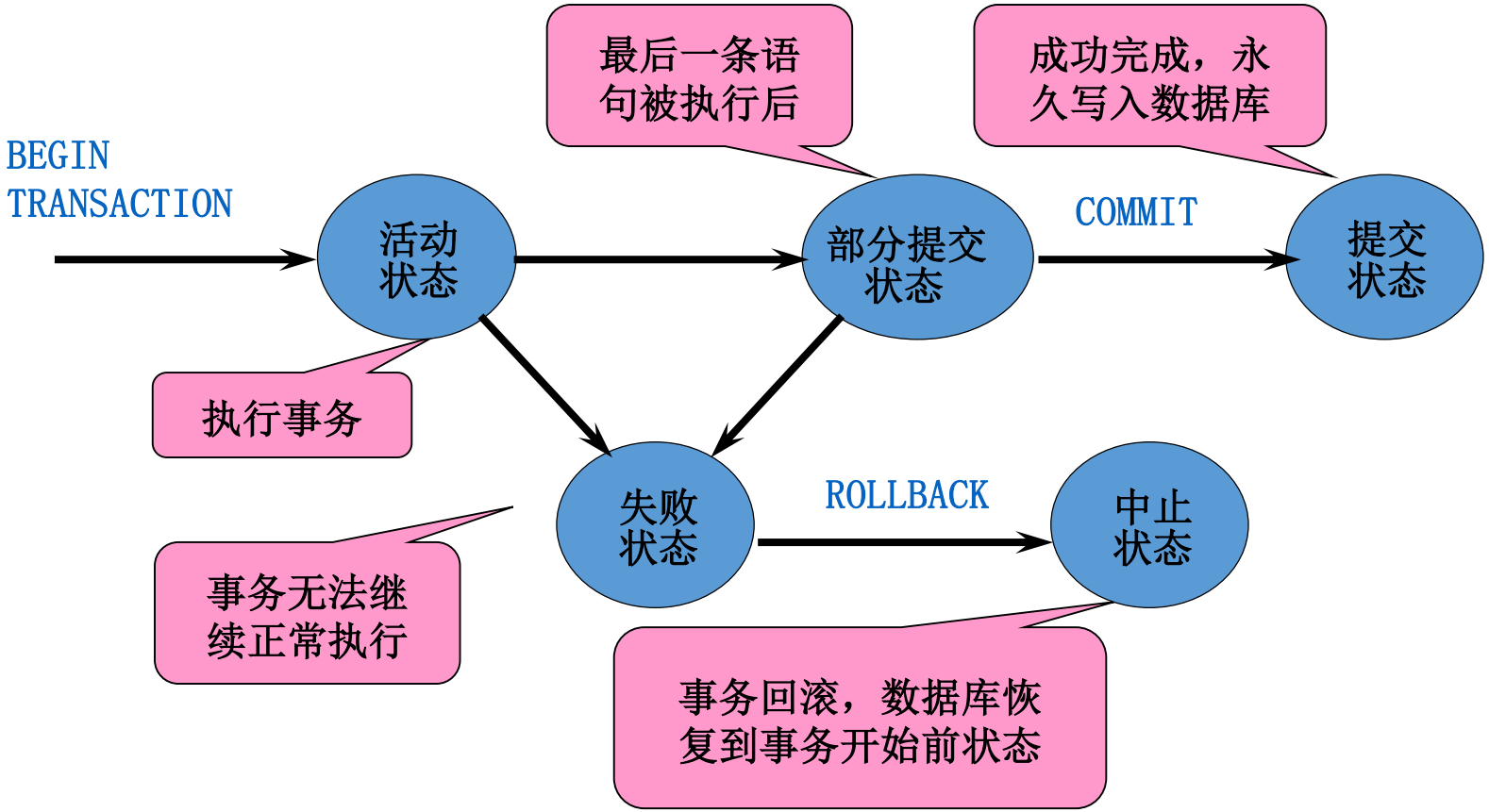
A. Normalize the above table to the 3 NF and draw the relational schema diagram and indicate the primary keys and the referential constraints.

B. Based on the above schema diagram draw the ER diagram.

- You have been asked to design a database for the university administration, which records the following information:
- 1. All students necessarily have a unique student ID, a name, and a university email address. Each student is also either an undergraduate or a graduate student.
- 2. Each graduate student has an advisor.
- 3. Each undergraduate student has a major.
- 4. Students take courses. A student may take one course, multiple courses, or no courses.
- 5. Each course has a course number, course name, and days of the week the course is scheduled.
- 6. Each course has exactly one head TA, who is a graduate student.
- 7. Every head TA has an office where he or she holds office hours.
- A. Draw an ER diagram for this application. Be sure to mark the multiplicity of each relationship of the diagram. Decide the key attributes and identify them on the diagram. Please state all assumptions you make in your answers.
- B. Translate your ER diagram into a relational schema. Select approaches that yield the fewest number of relations; merge relations where appropriate. Specify the key of each relation in your schema.

- 事务（*transaction)*：——保证数据完整性
  - 将一组语句作为一个单元执行
  - 必须拥有称为ACID的四个属性
    - 原子性（*Atomicity*）：事务必须是原子工作单元；对于其数据修改，要么全都执行，要么全都不执行。
    - 一致性（***Consistency***）：事务在完成时，必须使所有的数据都保持一致状态。
    - 隔离性（***Isolation***）：一个事务的执行不会被另一个事务干扰
    - 持久性（***Durability***）：事务完成之后，它对于系统的影响是永久性的。该修改即使出现系统故障也将一直保持。

3、事务的状态



事务定义语句与状态的关系

- 数据库引擎 提供：
  - 锁定设备，使事务保持隔离。
  - 记录设备，保证事务的持久性。即使服务器硬件、操作系统或数据库引擎 实例自身出现故障，该实例也可以在重新启动时使用事务日志，将所有未完成的事务自动地回滚到系统出现故障的点。
  - 事务管理特性，强制保持事务的原子性和一致性。事务启动之后，就必须成功完成，否则数据库引擎 实例将撤消该事务启动之后对数据所做的所有修改。

- 封锁及锁的类型
  - 封锁机制是并发控制的主要手段。
  - 封锁是使事务对它要操作的数据有一定的控制能力。
  - 封锁具有3个环节：
    - 第一个环节是申请加锁，即事务在操作前要对它欲使用的数据提出加锁请求；
    - 第二个环节是获得锁，即当条件成熟时，系统允许事务对数据加锁，从而事务获得数据的控制权；
    - 第三个环节是释放锁，即完成操作后事务放弃数据的控制权。为了达到封锁的目的，在使用时事务应选择合适的锁，并要遵从一定的封锁协议。

- 基本的封锁类型有两种：排它锁(Exclusive Locks，简称X锁)和共享锁(Share Locks，简称S锁)。
- 排它锁
  - 排它锁也称为独占锁或写锁。
  - 一旦事务T对数据对象A(可以是数据项、记录、数据集以至整个数据库)加上排它锁(X锁)，则只允许T读取和修改A，也不能再对A加任何类型的锁，直到T释放A上的锁为止。
  - 可见X封锁只允许一个事务独锁某个数据，具有排他性。
  - 排它 (X) 用于数据修改操作，例如 INSERT、UPDATE 或 DELETE。确保不会同时同一资源进行多重更新。
- 共享锁
  - 共享锁又称读锁。
  - 如果事务T对数据对象A加上共享锁(S锁)，其他事务对A只能再加S锁，不能加X锁，直到事务T释放A上的S锁为止。
  - 共享锁(S) 用于不更改或不更新数据的操作（只读操作），如 SELECT 语句。

- **8.5 并发调度的可串行性**

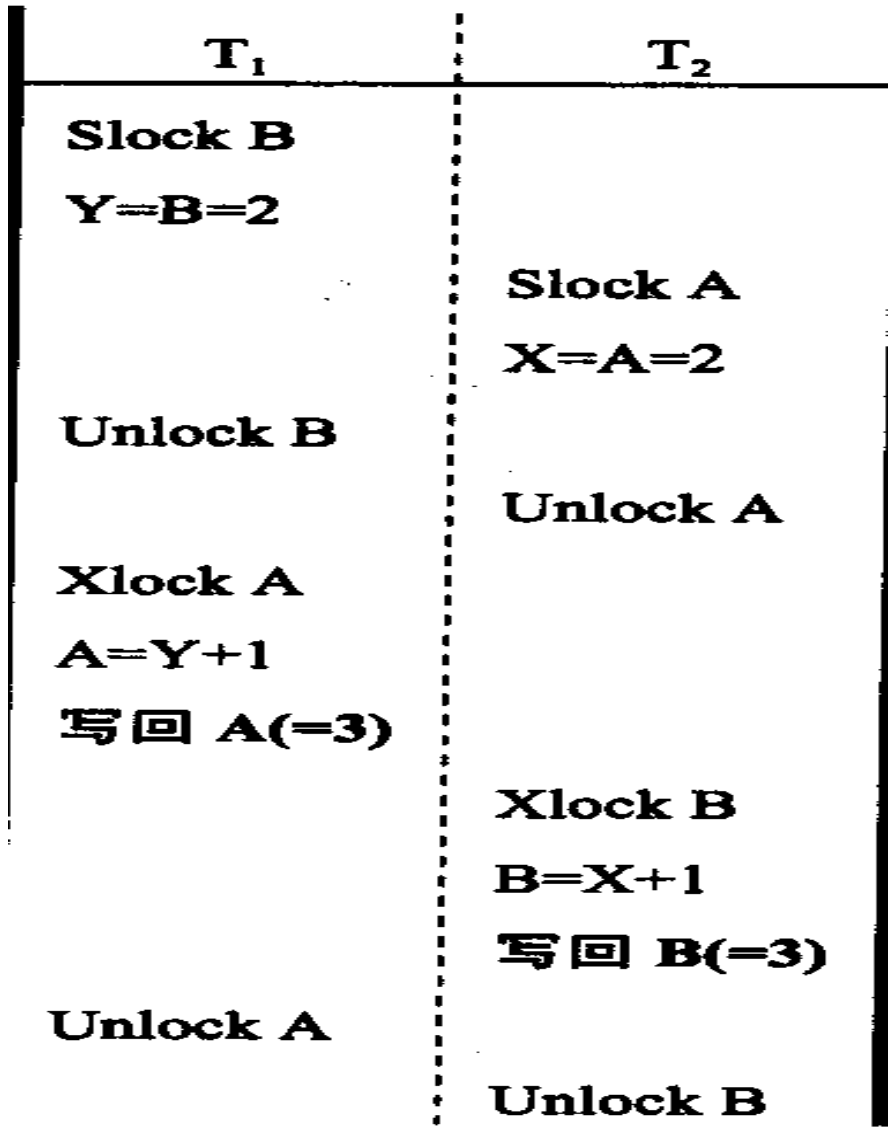
  - 计算机系统对并发事务中并发操作的调度是随机的，而不同的调度可能会产生不同的结果，那么哪个结果是正确的，哪个是不正确的呢？
  - 如果一个事务运行过程中没有其他事务同时运行，也就是说它没有受到其他事务的干扰，那么就可以认为该事务的运行结果是正常的或者预想的。
    - 因此将所有事务串行起来的调度策略一定是正确的调度策略。虽然以不同的顺序串行执行事务可能会产生不同的结果，但由于不会将数据库置于不一致状态，所以都是正确的。

T_1	T_2
Slock B	
Y=B=2	
Unlock B	
Xlock A	
A=Y+1	
写回 A(=3)	
Unlock A	
	Slock A
	X=A=3
	Unlock A
	Xlock B
	B=X+1
	写回 B(=4)
	Unlock B

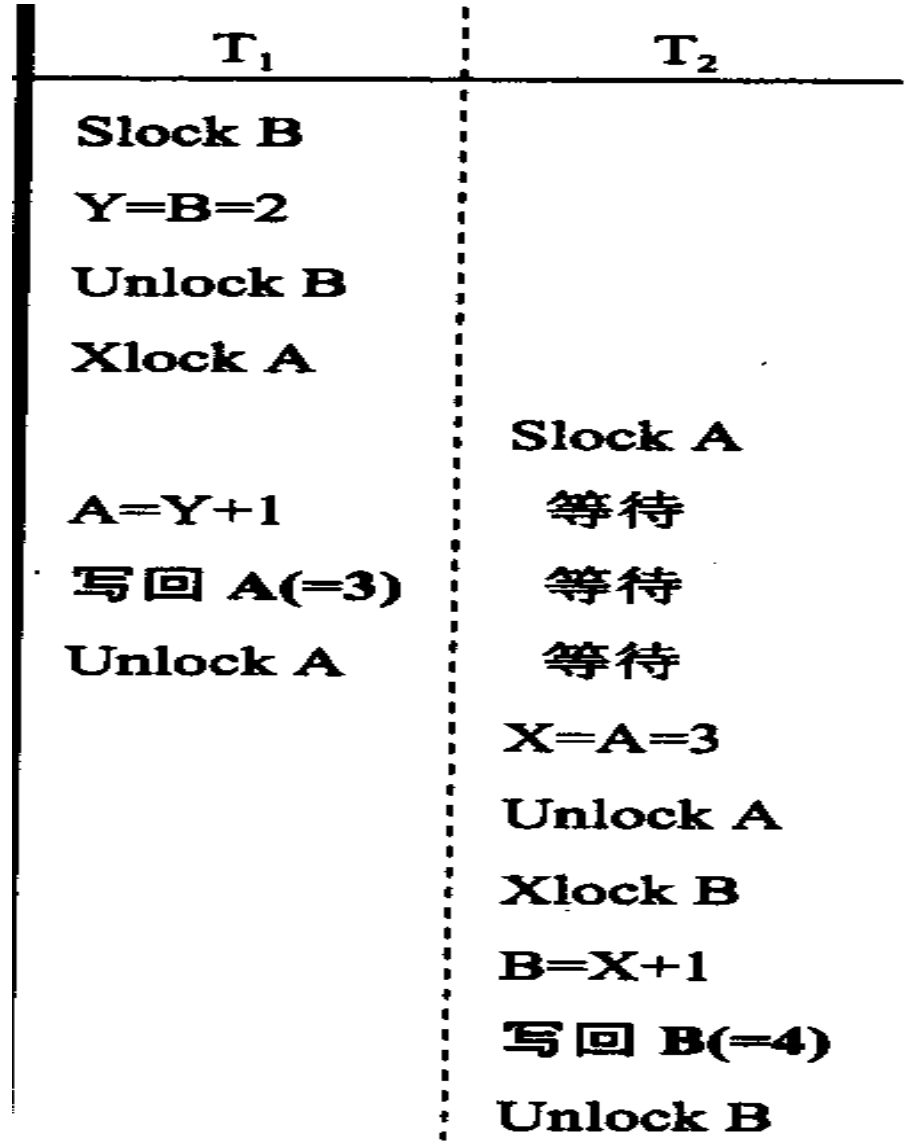
(a) 串行调度

T_1	T_2
	Slock A
	X=A=2
	Unlock A
	Xlock B
	B=X+1
	写回 B(=3)
	Unlock B
Slock B	
Y=B=3	
Unlock B	
Xlock A	
A=Y+1	
写回 A(=4)	
Unlock A	

(b) 串行调度



c) 不可串行化的调度



(d) 可串行化的调度

- **定义：** 多个事务的并发执行是正确的，当且仅当其结果与按某一次序串行地执行它们时的结果相同，我们称这种调度策略为可串行化（Serializable）的调度。

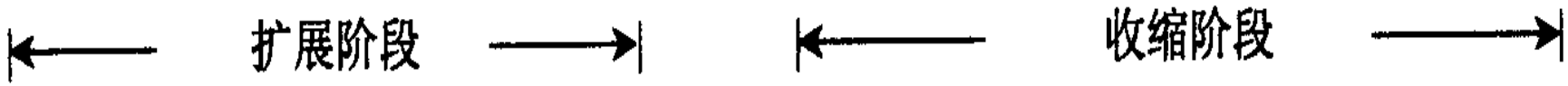
- **可串行性（Serializability）**是并发事务正确性的准则。按这个准则规定，一个给定的并发调度，当且仅当它是可串行化的，才认为是正确调度。

- 为了保证并发操作的正确性，DBMS的并发控制机制必须提供一定的手段来保证调度
  是可串行化的。
- 从理论上讲，在某一事务执行时禁止其他事务执行的调度策略一定是可串行化的调度，这也是最简单的调度策略，但这种方法实际上是不可取的，这使用户不能充分共享数据库资源。目前DBMS普遍采用封锁方法实现并发操作调度的可串行性，从而保证调度的正确性。
- **两段锁（Two-Phase Locking，2PL）协议**就是保证并发调度科串行化的封锁协议。

- **8.6 两段锁协议**
  - 所谓两段锁协议是指所有事务必须分两个阶段对数据项加锁和解锁。
    - 在对任何数据进行读、写操作之前，首先要申请并获得对该数据的封锁；
    - 在释放一个封锁之后，事务不再申请和获得任何其他封锁。
    - 所谓“两段”锁的含义是，事务分为两个阶段：
      - 第一阶段是获得封锁，也称为扩展阶段。在这阶段，事务可以申请获得任何数据项上的任何类型的锁，但是不能释放任何锁。
      - 第二阶段是释放封锁，也称为收缩阶段。在这阶段，事务可以释放任何数据项上的任何类型的锁，但是不能再申请任何锁。

例如事务 T_1 遵守两段锁协议，其封锁序列是：

Slock A　　Slock B　　Xlock C　　Unlock B　　Unlock A　　Unlock C；



又如事务 T_2 不遵守两段锁协议，其封锁序列是：

Slock A　Unlock A　Slock B　Xlock C　Unlock C　Unlock B；

可以证明，若并发执行的所有事务均遵守两段锁协议，则对这些事务的任何并发调度策略都是可串行化的。

T_1	T_2
Slock B	
读 B=2	
Y=B	
Xlock A	
	Slock A
	等待
A=Y+1	等待
写回 A=3	等待
Unlock B	等待
Unlock A	等待
	Slock A
	读 A=3
	Y=A
	Xlock B
	B=Y+1
	写回 B=4
	Unlock B
	Unlock A

(a) 遵守两段锁协议

T_1	T_2
Slock B	
读 B=2	
Y=B	
Unlock B	
Xlock A	
	Slock A
	等待
A=Y+1	等待
写回 A=3	等待
Unlock A	等待
	Slock A
	读 A=3
	X=A
	Unlock A
	Xlock B
	B=X+1
	写回 B=4
	Unlock B

(b) 不遵守两段锁协议

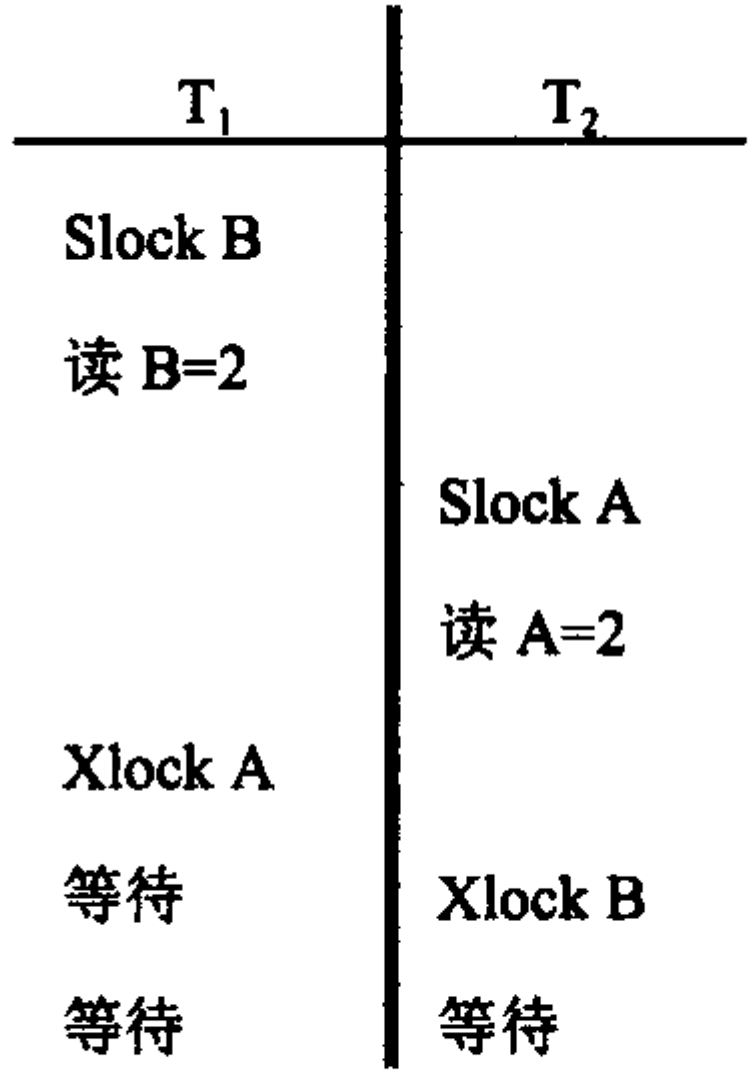


图 8.7　遵守两段锁协议的事务发生死锁

- 可以证明，若并发执行的所有事务均遵守两段锁协议，则对这些事务的任何并发调度策略都是可串行化的。

- 需要说明的是，事务遵守两段锁协议是可串行化调度的充分条件，而不是必要条件。也就是说；若并发事务都遵守两段锁协议，则对这些事务的任何并发调度策略都是可串行化的：若对并发事务的一个调度是可串行化的，不一定所有事务都符合两段锁协议。