



# 设计篇

## 第四章 数据库设计

主讲：高宏

海量数据计算研究中心





# Outline

- 数据库设计概述与需求分析
- 概念数据库设计
- 逻辑数据库设计
- 物理数据库设计
- 小结





## 4.1 数据库设计概述与需求分析





# 数据库设计

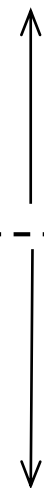
## • 数据库设计

- 对于一个给定的应用领域，设计优化的数据库逻辑和物理结构，使之满足用户的信息管理要求和数据操作要求，有效地支持各种应用系统的开发和运行
- 信息管理要求
  - 在数据库中应该存储和管理哪些数据对象。
- 数据操作要求
  - 对数据对象需要进行哪些操作，如查询、增、删、改、统计等操作。





# 数据库设计步骤



独立于数据库  
管理系统

与数据库  
管理系统  
相关

需求收集和分析

综合各个用户的应用需求(数据与处理)

设计概念结构

形成独立于机器特点, 独立于各个DBMS产品的概念模式(E-R图)

设计逻辑结构

将E-R图转换成具体的关系模型, 形成数据库逻辑模式

数据模型优化

根据用户处理的要求、安全性的考虑, 在基本表的基础上再建立必要的视图(View), 形成数据的外模式

设计物理结构

根据DBMS特点和处理的需要, 进行物理存储安排, 建立索引, 形成数据库内模式

性能评价预测

物理实现

利用具体DBMS系统建立数据库, 编写调试应用程序, 组织数据入库, 试运行

数据库运行维护

在数据库运行过程中, 不断对其评估、调整与修改





# 数据库设计

- 数据库设计一个主要部分：如何表示现实世界中各种类型的对象及对象间关系
- 容易出现的问题

## — 冗余

- 出现重复信息，为更新维护带来困难

Sno	Sdept	Cno	Cname	Grade
200215121	CS	1	数据库	94
200215122	CS	1	数据库	89
200215124	CS	1	数据库	91
200215126	CS	1	数据库	70
200215131	CS	1	数据库	88
200210103	MA	2	数据结构	87
200210106	MA	2	数据结构	90

## — 不完整

- 例：如果上面关系中缺少学生信息



# 需求分析

- 需求分析就是分析用户的要求
  - 是设计数据库的起点
  - 结果是否准确地反映了用户的实际要求，将直接影响到后面各个阶段的设计，并影响到设计结果是否合理和实用





# 需求分析

- 需求分析的重点是

- “数据”和“处理”，获得用户对数据库要求

- 信息要求

- 用户需要从数据库中获得信息的内容与性质
- 由用户的信息要求可以导出数据要求，即在数据库中需要存储哪些数据

- 处理要求

- 用户要完成什么处理功能
- 对处理的响应时间的要求
- 对处理方式的要求(批处理 / 联机处理)

- 安全性与完整性要求







## 4.2 概念数据库设计





# Outline

- 概述
- 实体联系模型
- 事务的设计





## 4.2.1 概述

- 概念数据库设计的任务包括两方面

- 概念数据库模式设计

- 以需求分析阶段所识别的数据项和应用领域的未来改变信息为基础，使用高级数据模型建立概念数据库模式
    - 概念数据库模式独立于任何数据库管理系统，不能直接用于数据库的实现。
    - 数据模型：实体联系模型

- 事务设计

考察需求分析阶段提出的数据库操作任务，形成数据库事务的高级说明





# Outline

- 概述
- 实体联系模型
- 事务的设计





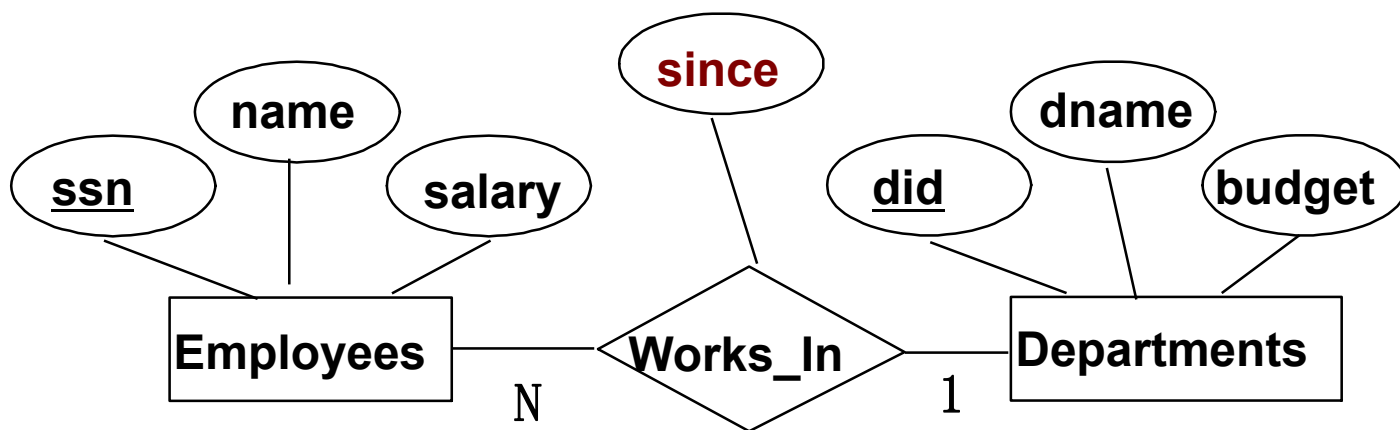
- 实体联系模型

- Entity-Relationship model, 简称ER模型

- 表示成“实体-联系”图

- 三个主要元素

- 实体
- 属性
- 联系





## 4.2.2 实体联系模型

### • 基本概念

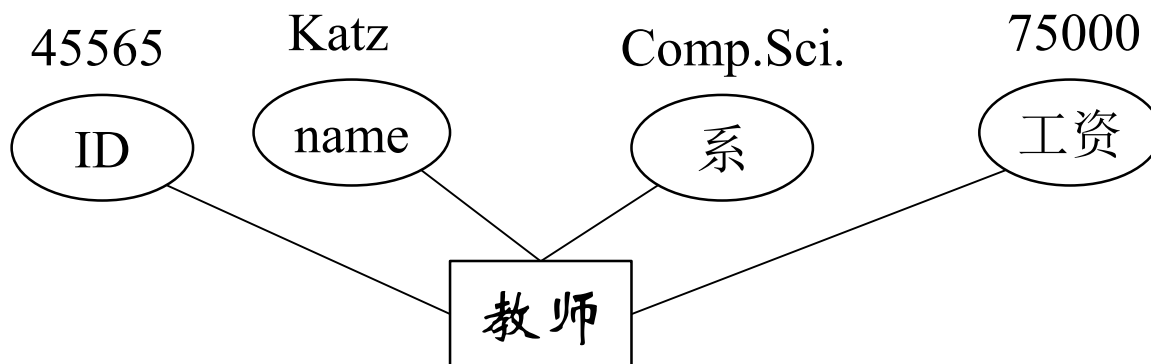
- **实体**是ER模型的基本对象。实体是现实世界中各种事物的抽象。
  - 实体可以是物理存在的事物，如人、汽车；
  - 也可以是抽象的概念，如学校、课程。
- 每个实体都有一组特征或性质，称为实体的**属性**。实体属性的一组特定值确定了一个特定的实体。**实体的属性值**是数据库中存储的主要数据。
  - 例如，学生实体具有名字、年龄、性别等属性





## • 基本概念(续)

### — 实体例子





- 基本概念(续)

- 实体集

- 是相同类型(即具有相同性质或属性)的实体集合
      - 例如, 某个大学所有教师的集合可被定义为实体集 *Teacher*.
    - 实体集不必互不相交







ID	name	dept_name	salary
76766	Crick	Biology	72000
45565	Katz	Comp. Sci.	75000
10101	Srinivasan	Comp. Sci.	65000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000
12121	Wu	Finance	90000
76543	Singh	Finance	80000
32343	El Said	History	60000
58583	Califieri	History	62000
15151	Mozart	Music	40000
33456	Gold	Physics	87000
22222	Einstein	Physics	95000

实体集: *Teacher, student*

学号 Sno	姓名 Sname	性别 Ssex	年龄 Sage	所在系 Sdept
200215121	李勇	男	20	CS
200215122	刘晨	女	19	CS
200215123	王敏	女	18	MA

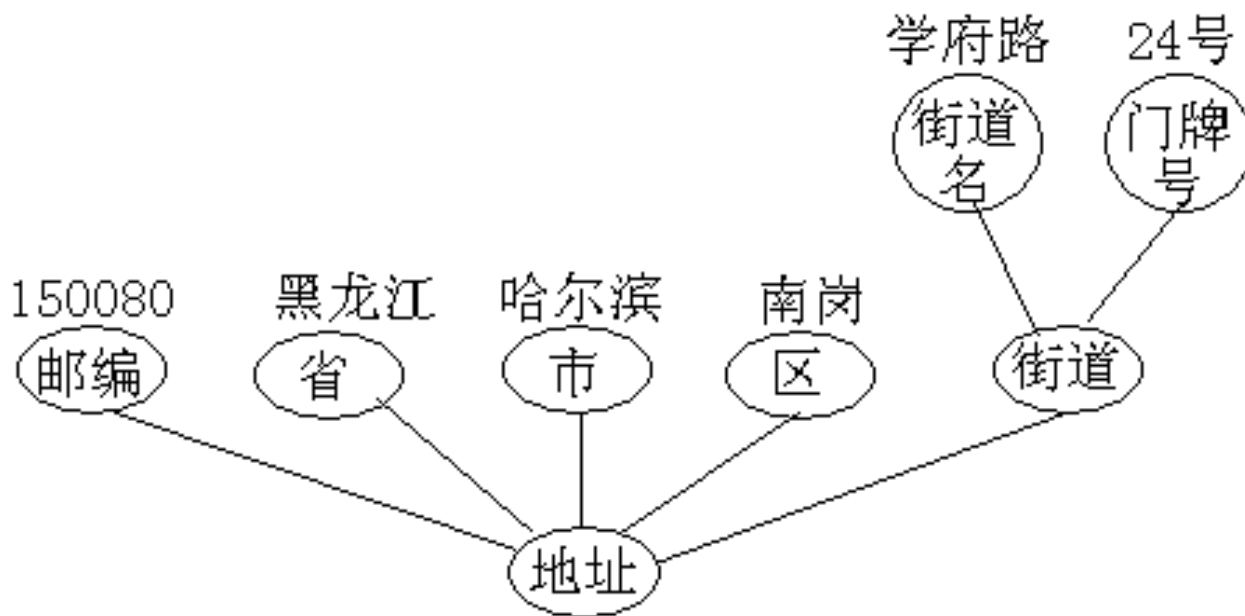




## • 基本概念(续)

### — 实体属性

- 简单属性：不能划分为更小部分
- 复合属性：可以划分为多个具有独立意义的子属性。复合属性具有层次结构





## • 基本概念(续)

### — 实体属性

- **单值属性：**对于同一个实体只能取一个值。
  - 例如，同一个人只能具有一个年龄，所以人的年龄属性是一个单值属性。
- **多值属性：**在某些情况下，实体的一些属性可能取多个值。
  - 例如：职务、联系方式等





## • 基本概念(续)

### — 实体属性

#### • 导出属性 (派生属性, derived attribute)

- 从当前日期和生日属性的值可以确定年龄属性的值, 即年龄属性的值可以由其他属性导出。我们称这样的属性为**导出属性**。导出属性的值不仅可以从另外的属性导出, 也可以从有关的实体导出。例如, 一个公司实体的雇员数属性的值可以通过累计该公司所有雇员数得到。

#### • 空值

- 在某些情况下, 实体的有些属性可能没有适当值可设置。这些属性通常被设置一个称为**空值**的特殊值。
- 例如, 一个未获得任何学位的人的学位属性只能被设置为空值





- 基本概念(续)

- 码：在ER模型中用来区分给定实体集中不同的实体
- 关系模式中关于码的概念直接适用于实体集
  - 超码
  - 候选码
  - 主码





## • 基本概念(续)

### — 联系

- 不同实体之间可能具有某种关联，我们称这种关联为实体间的联系

— 例如，“1837101”与“CS”、“高宏”和“数据库系统”

### — 联系集

- 相同类型联系的集合

— 例如，学生和院系；教师与课程

- 联系集的度

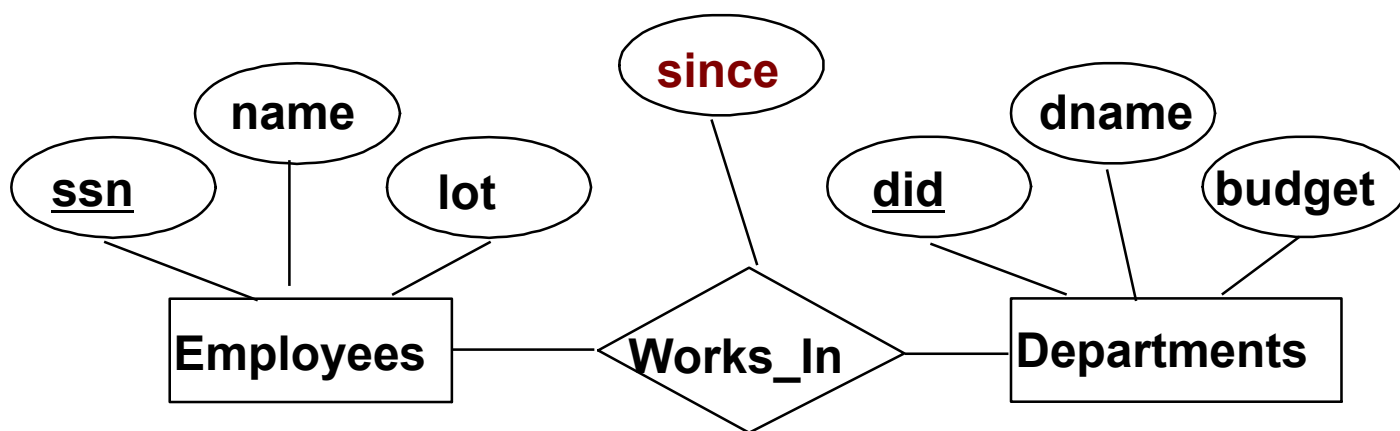
— 参与联系集的实体集的数目





## • 基本概念(续)

— 联系集的属性：描述性属性



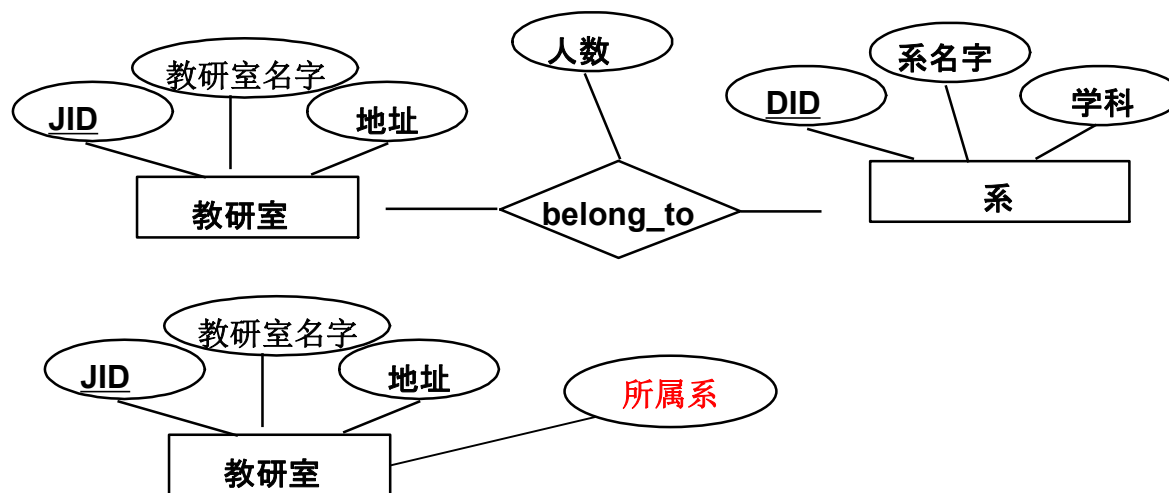


## • 基本概念(续)

— 实体之间的联系既可以使用联系集定义，也可以通过实体属性来表示。

• 例如，可以用属性表示实体教研室和系之间的所属联系。

— 用教研室实体的属性“所属系”的值来表示这个教研室实体所属的系实体。







- 基本概念(续)

- 映射基数(Mapping Cardinality)

- 一个实体通过联系集能关联的实体的个数

- 映射基数决定了实体之间存在3种联系

- 一对一

- 一个系只有一个系主任，一个大学只有一个校长

- 一对多（或多对一）

- 一个系有很多学生

- 多对多

- 一个学生可选修多门课程，每门课程可被多个学生选修





## • 基本概念(续)

### — 弱实体

- 现实世界中存在这样的一些实体集，它们没有足够的属性形成自己的**主码**。为了区分各个实体，它们必须与其它实体集相关联。这样的实体集称为弱实体集合
  - 弱实体集中的不同实体的属性可能完全相同
- 与弱实体集相关联的实体集：主实体集
- 主实体集与它的弱实体集之间的联系称为识别联系。
- 主实体与弱实体的关系都为**一对多**的关系，弱实体为**多方**。



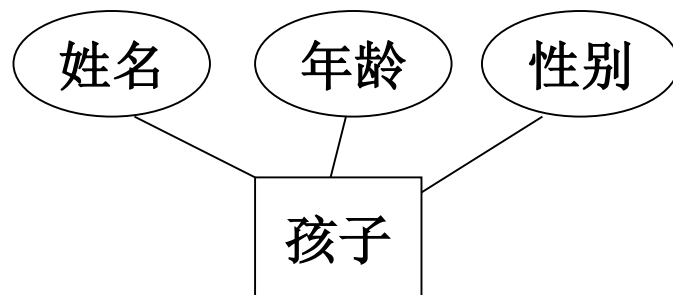


## • 基本概念(续)

### — 弱实体

#### • 例：父亲实体集与孩子实体集

- 不同父亲的孩子可以具有相同的姓名、年龄和性别
- 同一个父亲的孩子一定具有不同的名字
- 显然，孩子实体集是弱实体集





## • 基本概念(续)

### — 弱实体

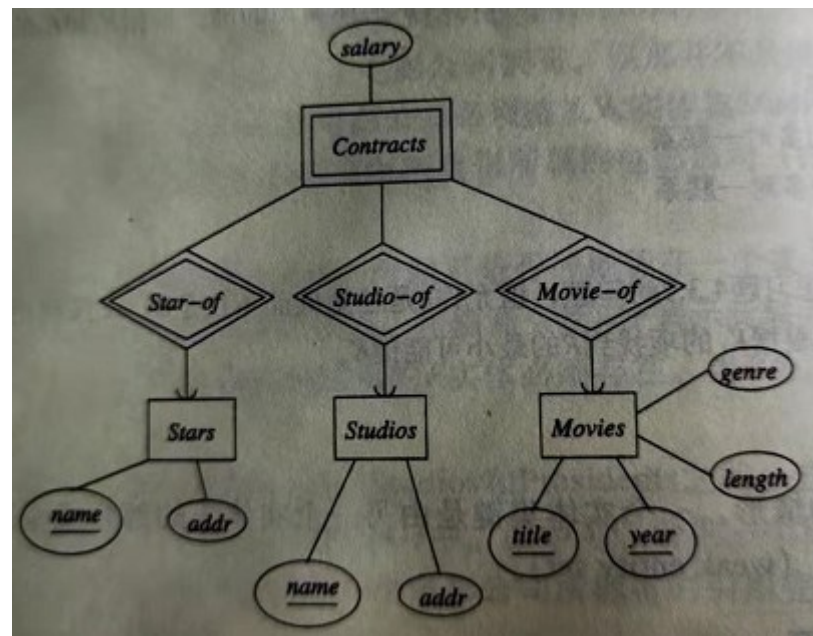
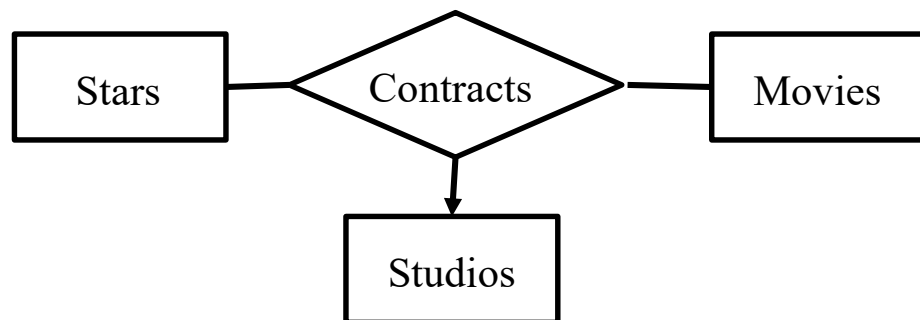
- 弱实体集必须具有一个或多个属性，使得这些属性可以与主实体集的主码相结合，形成相应弱实体集的主码
- 这样的弱实体属性称为弱实体集的部分码
  - 给定一个弱实体集，可以使用它的主实体集的码和它的部分码识别不同的弱实体。
  - 例如，上例中的孩子名是弱实体集孩子的部分码





## • 基本概念(续)

### — 弱实体



实体集Contracts只有一个属性salary，它无法做主键





- 扩展ER模型
  - ISA联系
- UML（统一建模语言）



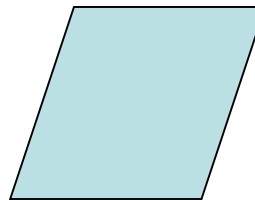
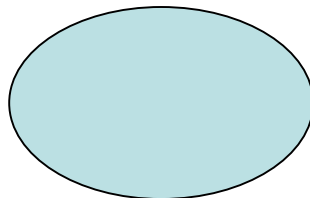


## • 实体联系图

— 是表示ER模型的图形工具，简称**ER图**


— ER图用来表示实体集和实体联系集

- 矩形：表示实体集。
- 椭圆：表示属性。
- 菱形：表示联系集。
- 线段：将属性连接到实体集或将实体集连接到联系集。




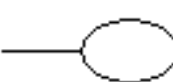


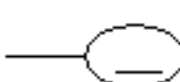
1. 实体集: 

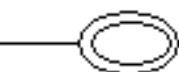
2. 弱实体集: 

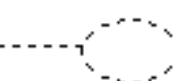
3. 联系集: 

4. 识别联系集: 


5. 属性: 


6. 键属性: 


7. 多值属性: 


8. 导出属性: 

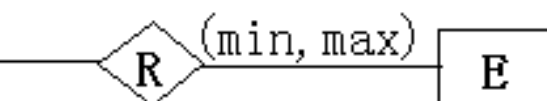
8. 复合属性: 

10. B具有全域实体关联约束: 

11. 1:1联系集: 

12. 1:N联系集: 

13. M:N联系集: 

14. 实体集 E 上的结构约束: 

# ER 图 中 基 本 符 号



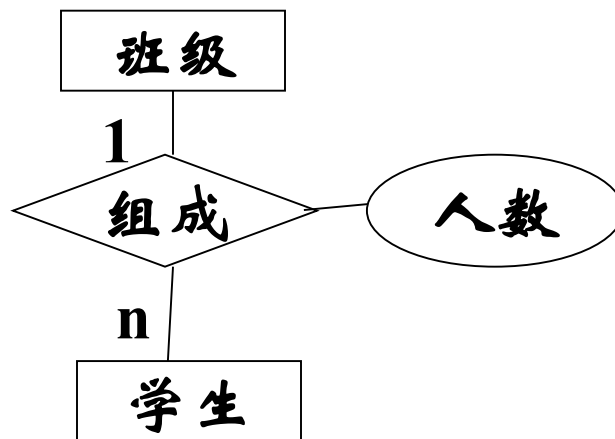
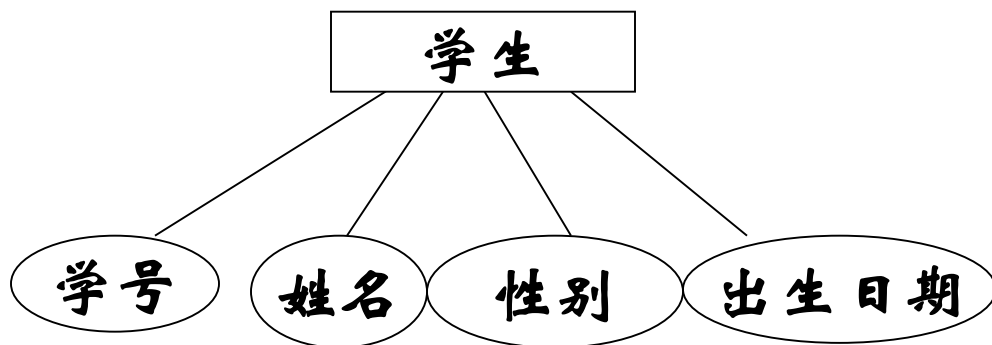




实体名

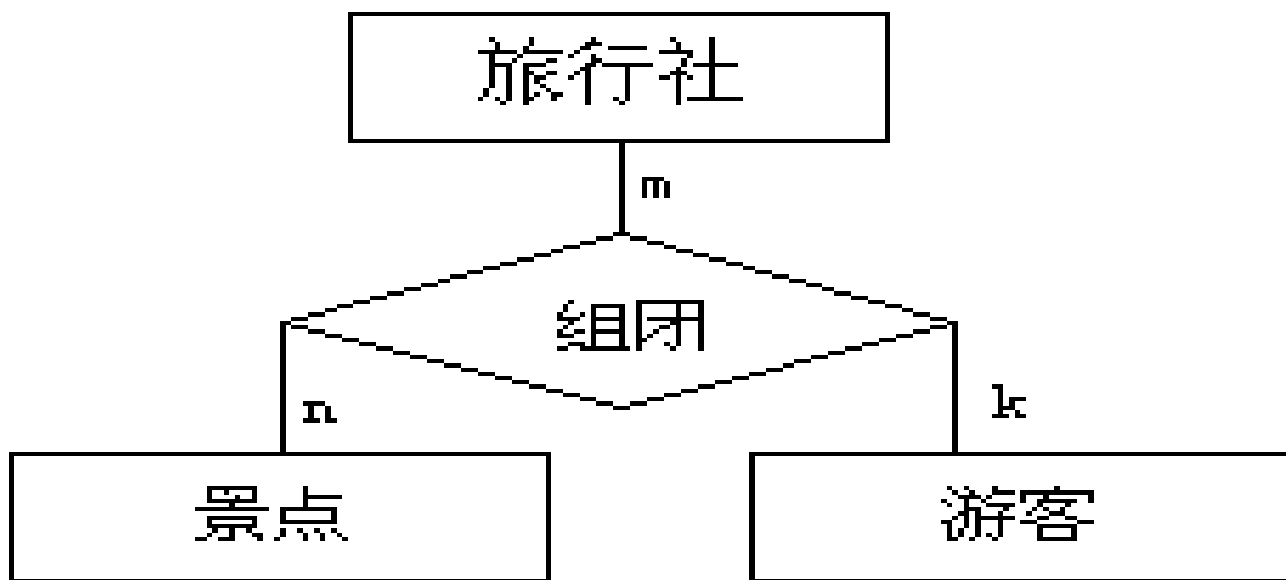
属性名

联系名





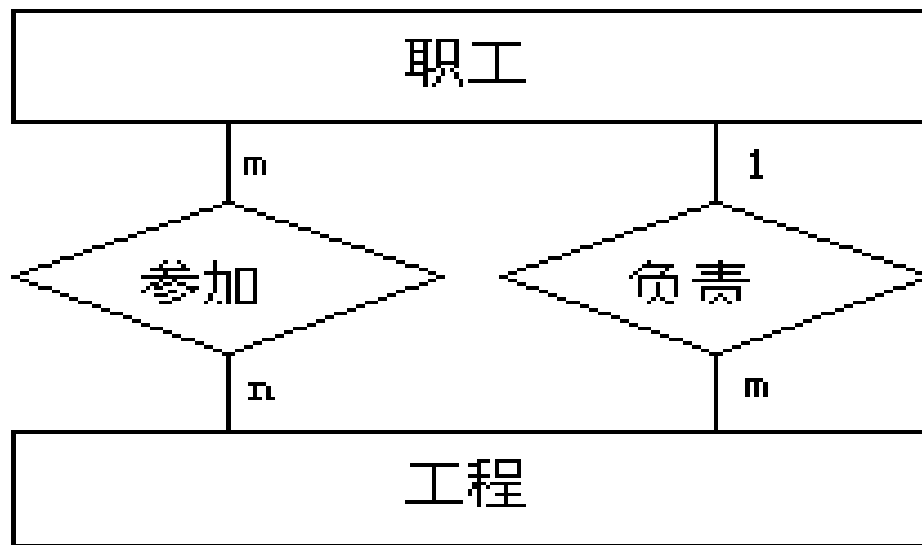
- 两个不同实体集间的联系
- 两个以上实体间的多元联系





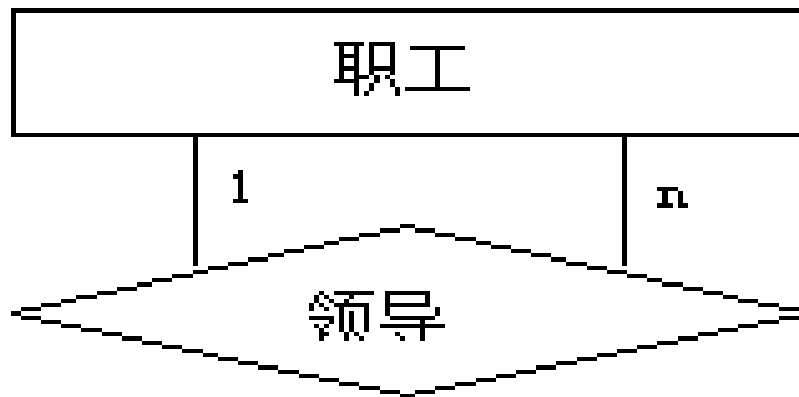
## • 两个不同实体集间的多种联系

- 职工与工程间，一个职工可以参加多个工程，一个工程可以有多个职工参加，同时一个工程由一个职工负责，一个职工可以负责多个工程





- 同一实体内部个体间的二元联系





## • ER模型设计实例

根据财经学院数据库应用需求，运用E-R模型的基本方法设计财经学院教学数据库的概念模型。

1、根据需求调查，初步确定所关心的数据对象：

学院、系、教师、班级、学生、课程、成绩

2、根据业务规则，设计初步E-R模型

(1) 学院有多个系，每个系只能属于一个学院。

(2) 每个系有多个班级，而每个班级只能属于一个系。

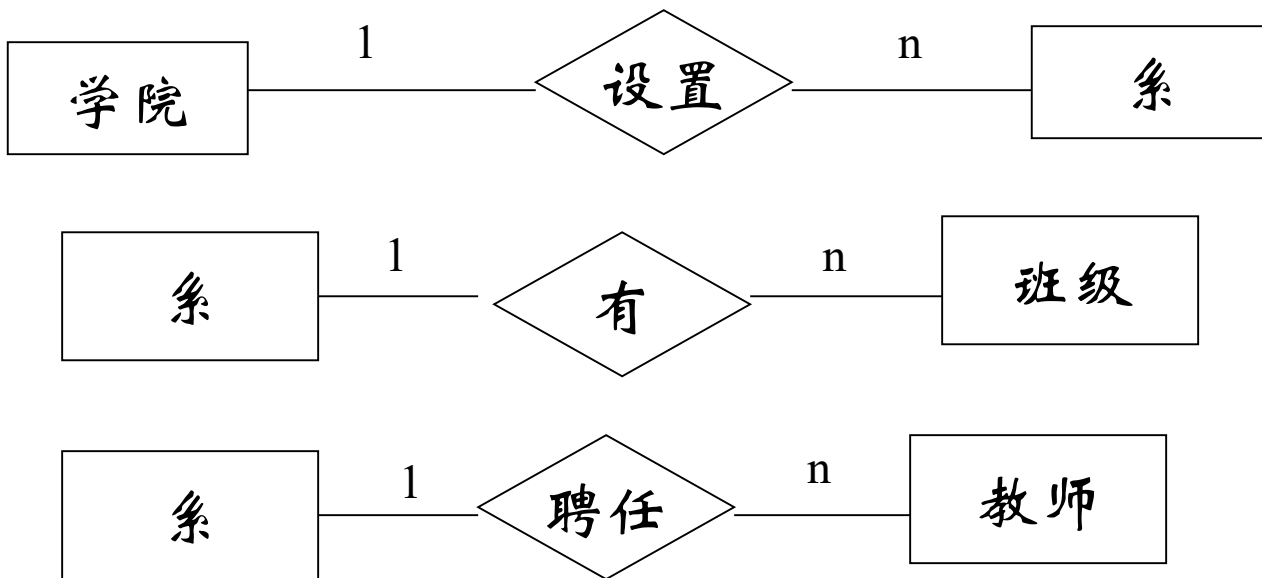
(3) 每个系聘任多名教师，而每个教师又只能属于一个系。





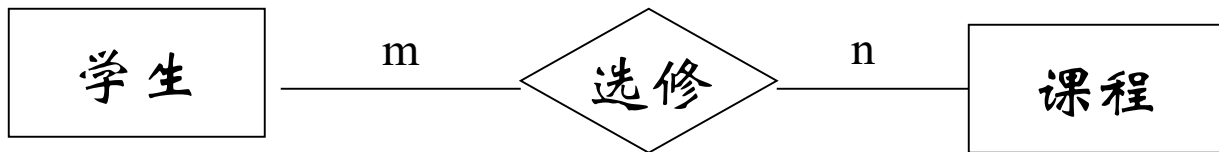
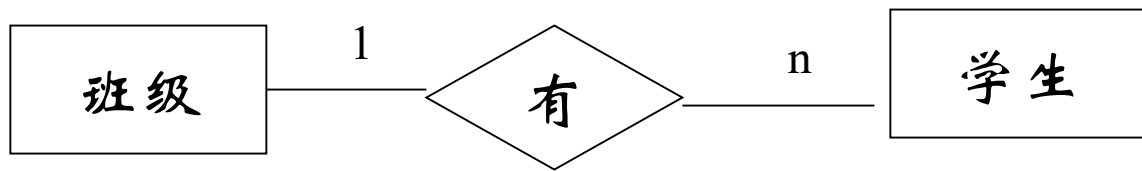
- (1) 学院有多个系，每个系只能属于一个学院。
- (2) 每个系有多个班级，而每个班级只能属于一个系。
- (3) 每个系聘任多名教师，而每个教师又只能属于一个系

归纳上述6项可定义6个实体:学院、系、班级、教师、学生、课程。



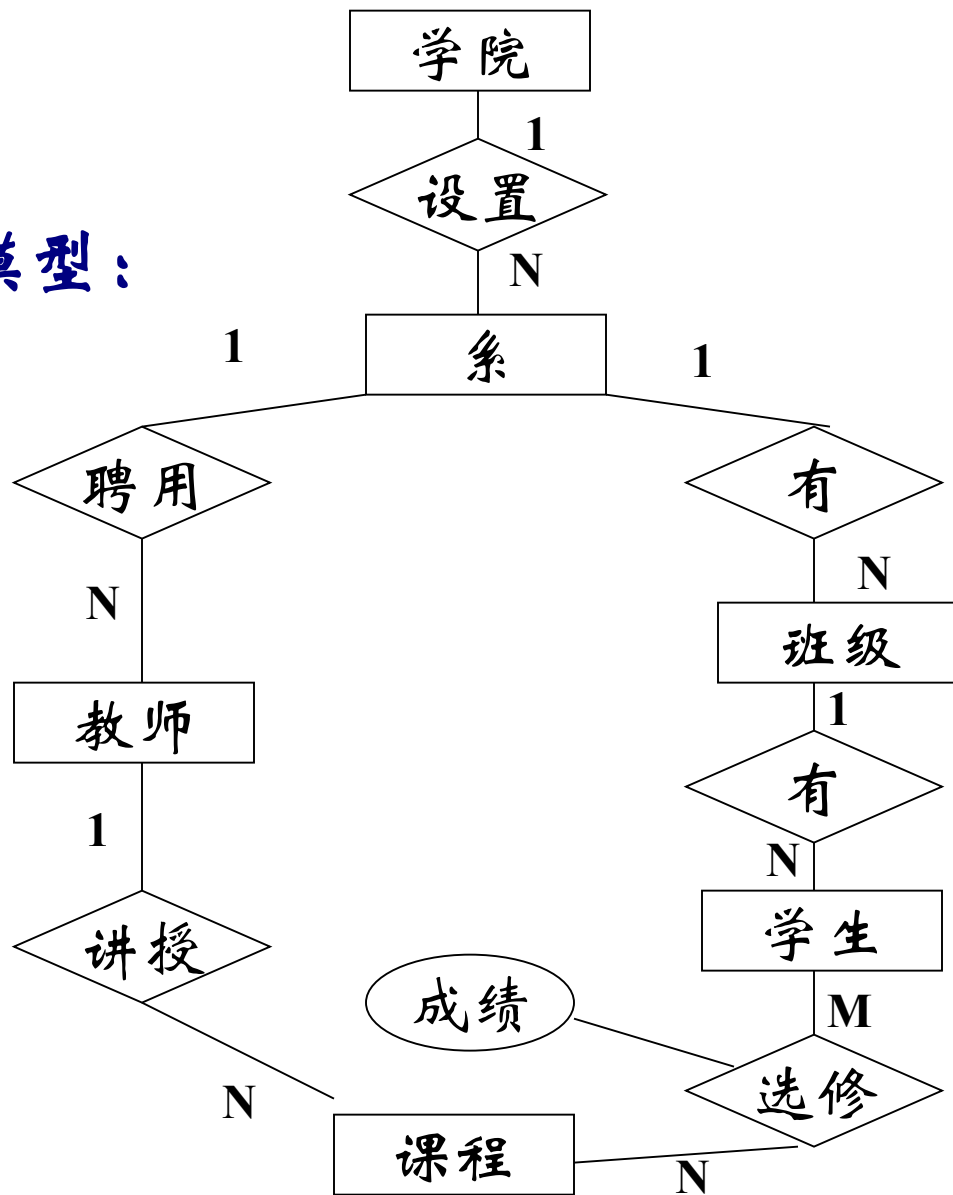


- (1) 学院有多个系，每个系只能属于一个学院。
- (2) 每个系有多个班级，而每个班级只能属于一个系。
- (3) 每个系聘任多名教师，而每个教师又只能属于一个系





## 整体E-R模型：





# • 实例

- 用E-R图来表示某个工厂物资管理的概念模型
- 物资管理涉及的实体有
  - 仓库 属性有仓库号、面积、电话号码
  - 零件 属性有零件号、名称、规格、单价、描述
  - 供应商 属性有供应商号、姓名、地址、电话号码、账号
  - 项目 属性有项目号、预算、开工日期
  - 职工 属性有职工号、姓名、年龄、职称

这些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，因此仓库和零件具有多对多的联系。用库存量来表示某种零件在某个仓库中的数量。

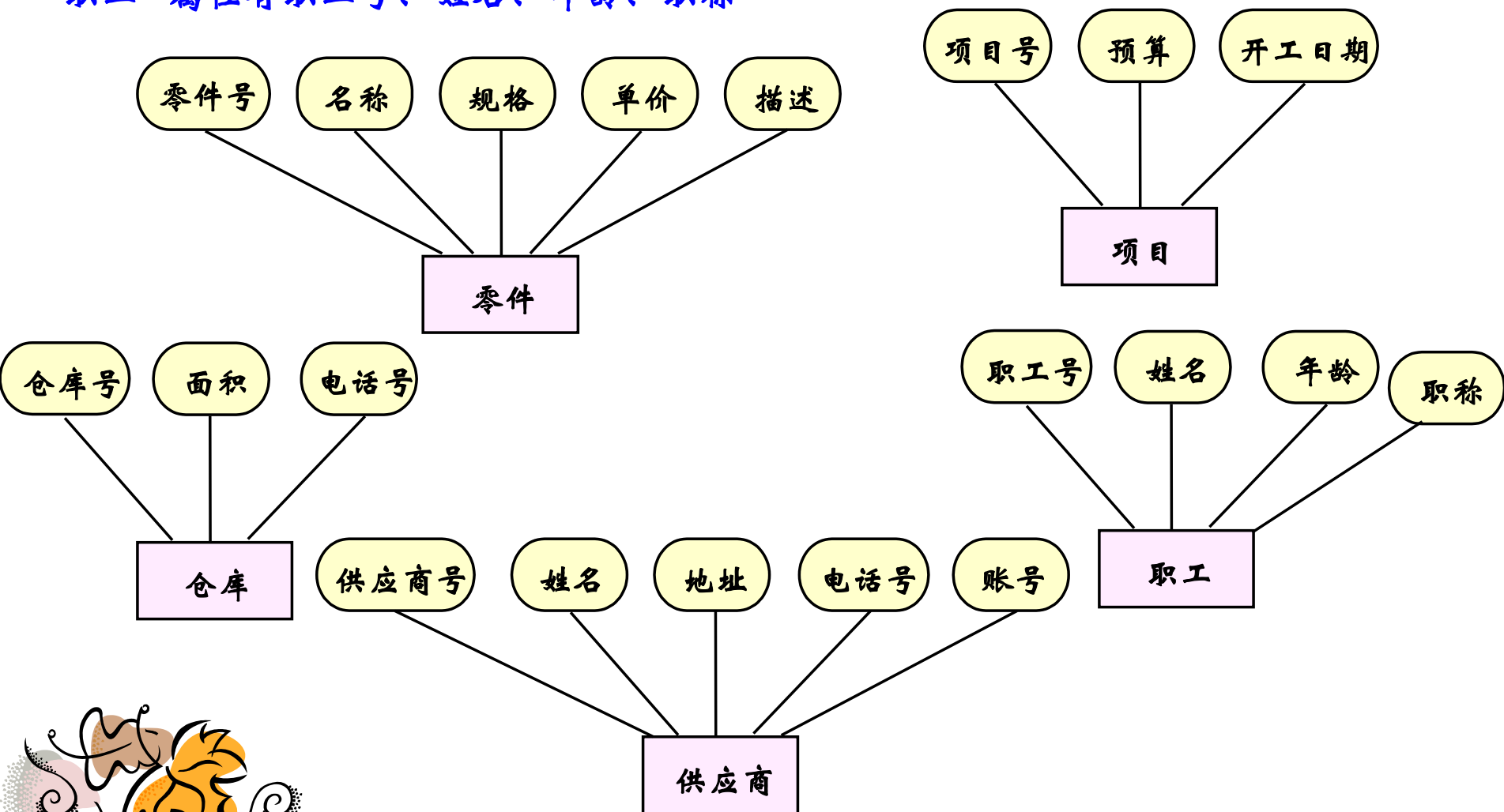
(2) 一个仓库有多个职工当仓库保管员，一个职工只能在一个仓库工作，因此仓库和职工之间是一对多的联系

(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，因此职工实体集中具有一对多的联系。

(4) 供应商、项目和零件三者之间具有多对多的联系。即一个供应商可以供给若干项目多种零件，每个项目可以使用不同供应商供应的零件，每种零件可由不同供应商供给。



- 仓库 属性有仓库号、面积、电话号码
- 零件 属性有零件号、名称、规格、单价、描述
- 供应商 属性有供应商号、姓名、地址、电话号码、账号
- 项目 属性有项目号、预算、开工日期
- 职工 属性有职工号、姓名、年龄、职称



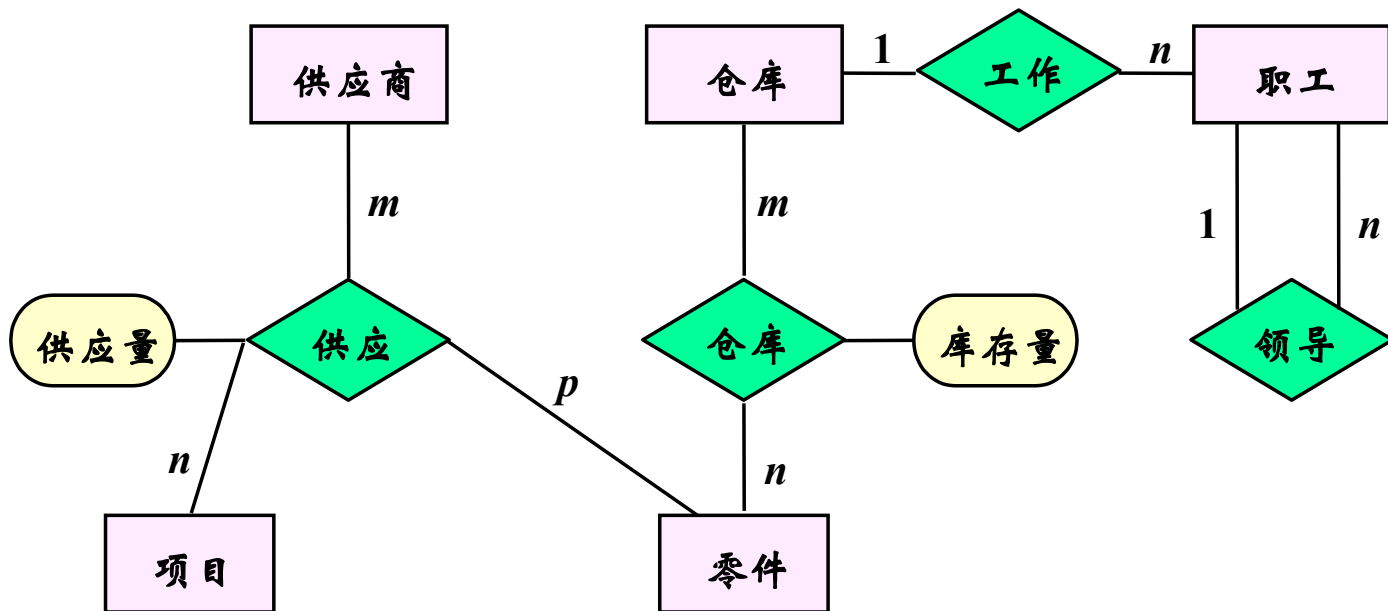
(a) 实体及其属性图

## 实体之间的联系:

- (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, 因此仓库和零件具有多对多的联系。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, 因此仓库和职工之间是一对多的联系

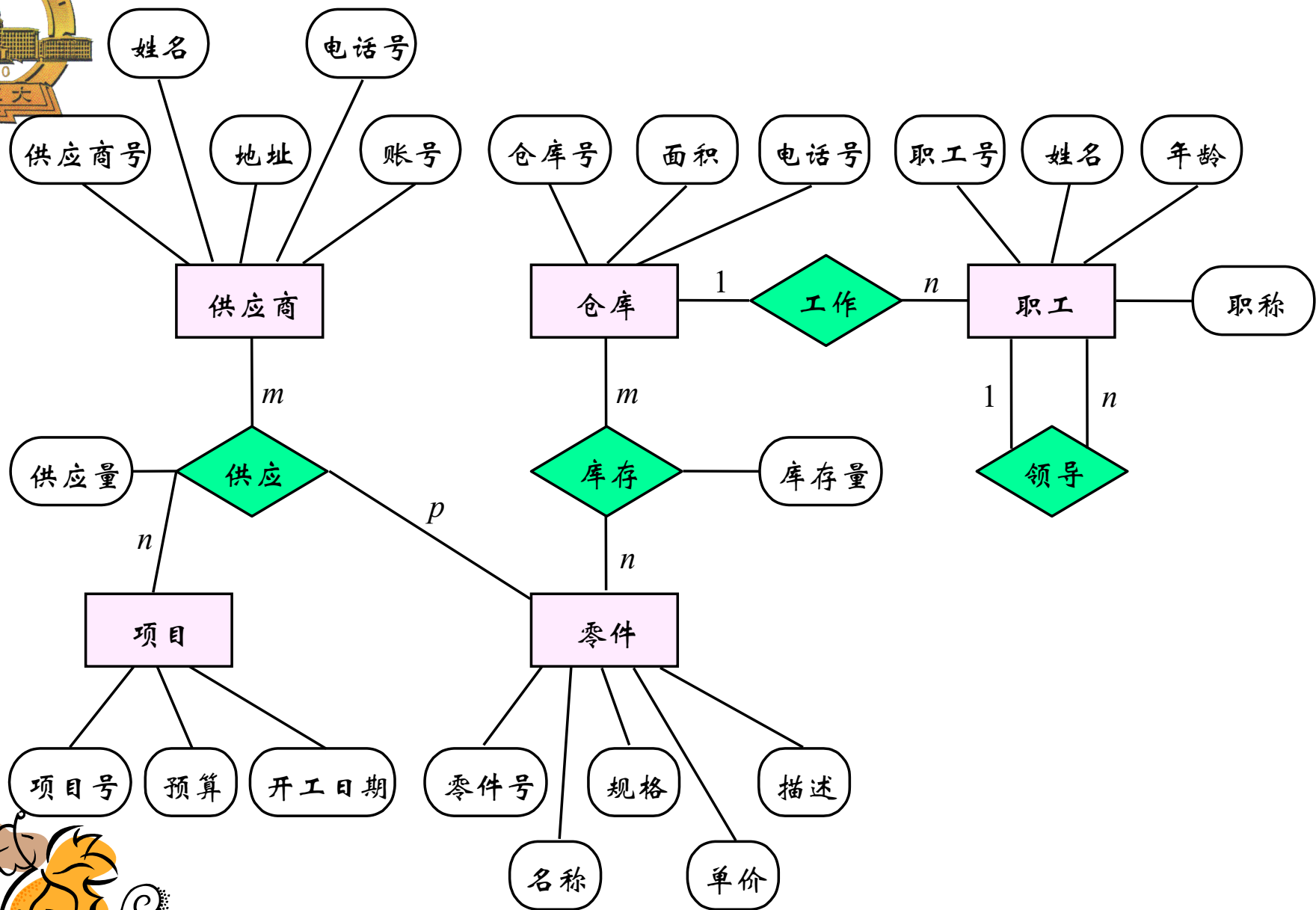
(3) 职工之间具有领导-被领导关系。即仓库主任领导若干保管员, 因此职工实体集中具有一对多的联系。

- (4) 供应商、项目和零件三者之间具有多对多的联系。即一个供应商可以供给若干项目多种零件, 每个项目可以使用不同供应商供应的零件, 每种零件可由不同供应商供给。



(b) 实体及其联系图





(c) 完整的实体联系图



# Outline

- 概述
- 实体联系模型
- 事务的设计





# 事务的设计

- 事务

- 一个或多个数据操作构成的集合，这组操作满足原子性。

- 例如，银行从账户A到账户B的一次资金转账操作

- 事务设计任务：定义事务功能

- 说明事务的输入、输出





## 4.3 逻辑数据库设计





# 逻辑数据库设计

- 逻辑数据库设计的任务

- 把概念数据库设计阶段产生的概念数据库模式  
变换为逻辑数据库模式

- 逻辑数据库设计的步骤

- 形成初始关系数据库模式
- 函数依赖与关系模式规范化
- 关系模式优化
- 定义关系上的完整性和安全性约束
- 子模式定义
- 性能估计





# 逻辑数据库设计

- 逻辑数据库设计的步骤

- 形成初始关系数据库模式

- 函数依赖与关系模式规范化

- 关系模式优化

- 定义关系上的完整性和安全性约束

- 子模式定义

- 性能估计





## 4.3.1 形成初始关系数据库模式

- 初始关系数据库模式是指直接由概念数据库模式生成的关系数据库模式。
- 初始关系数据库模式生成的目的是把概念数据库模式的实体、实体间联系等模型结构变换为关系模式。





## 4.3.1 形成初始关系数据库模式

- 由概念数据库模式生成初始关系数据库模式的方法：
  - 普通实体集的变换
  - 弱实体的变换
  - 多值属性的变换
  - 实体间联系的变换
  - 确定函数依赖集





## 4.3.1 形成初始关系数据库模式

- 普通实体集的变换

- 为概念数据库模式中的每个普通实体集 $E$ 建立一个关系 $S$ 。
- $S$ 包含 $E$ 的所有简单属性和 $E$ 的复合属性的简单子属性。
- $E$ 的主码是 $S$ 的主码。



教研室关系G(名字, 编号, 地点, 所属系)

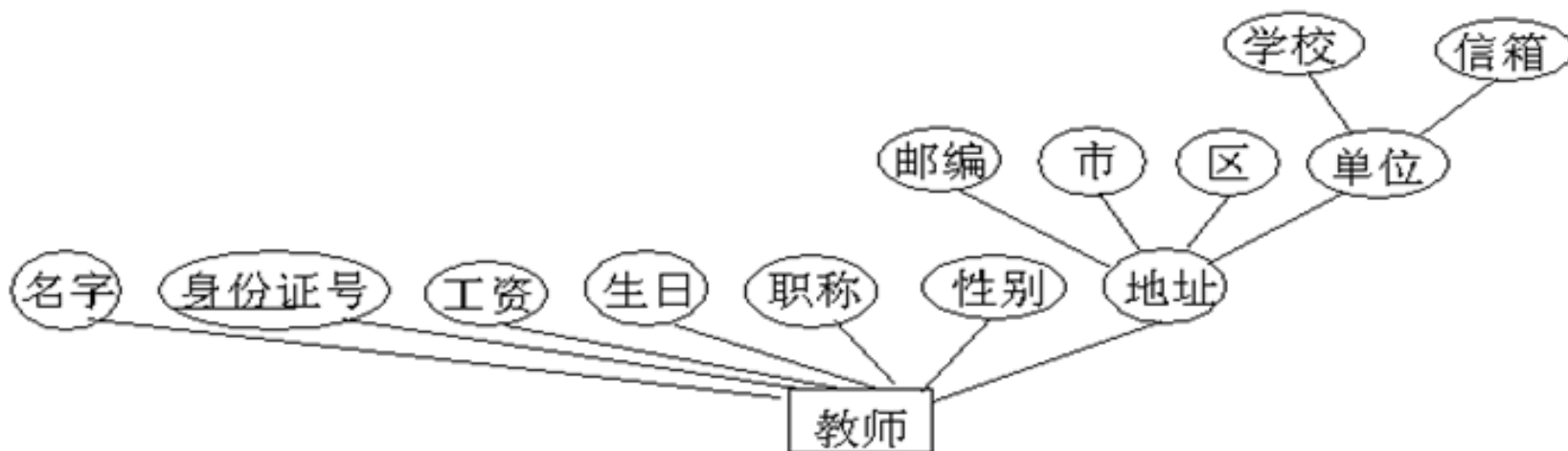




## 4.3.1 形成初始关系数据库模式

- 普通实体集的变换

— 例



教师关系T(名字, 身份证号, 工资, 生日, 职称, 性别, 邮编, 市, 区, 学校, 信箱)



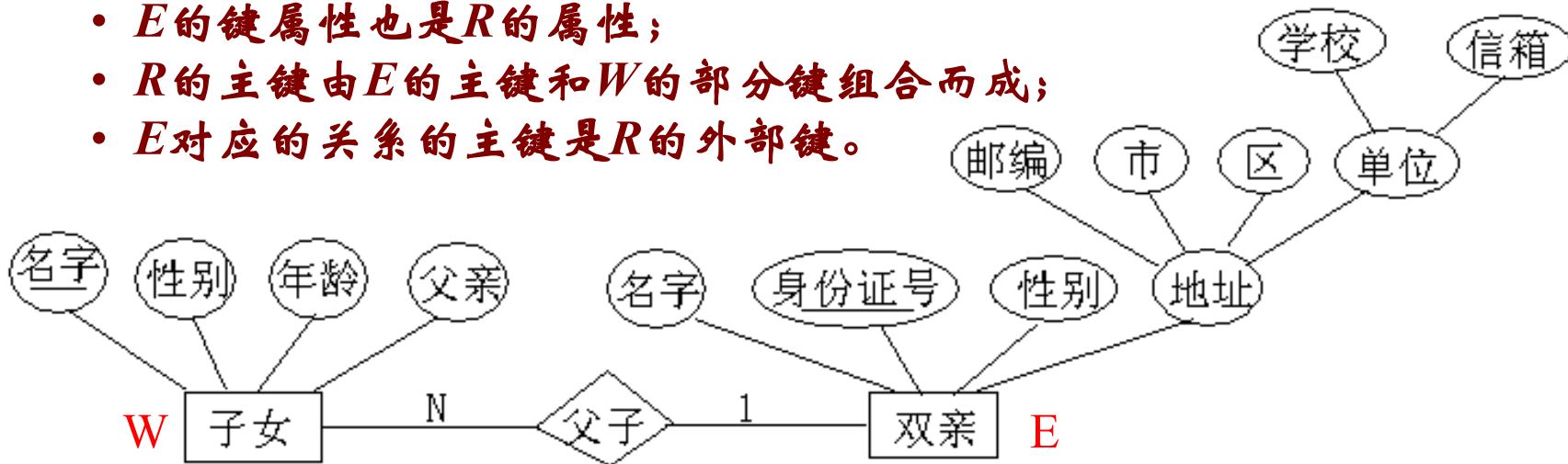


## 4.3.1 形成初始关系数据库模式

### • 弱实体的变换

— 设  $W$  是概念数据库模式中以实体集  $E$  为识别实体集的弱实体。

- 建立一个与  $W$  对应的关系  $R$ ;
- $W$  的所有简单属性和复合属性的简单子属性映射为  $R$  的属性;
- $E$  的键属性也是  $R$  的属性;
- $R$  的主键由  $E$  的主键和  $W$  的部分键组合而成;
- $E$  对应的关系的主键是  $R$  的外部键。



由弱实体“子女”和识别实体“双亲”  
可以生成关系  $R$   
(身份证号, 名字, 性别, 年龄, 父亲)

识别实体“双亲”可以生成关系  $T$   
(身份证号, 名字, 性别, 邮编, 市, 区,  
学校, 信箱)



## 4.3.1 形成初始关系数据库模式

### • 多值属性的变换

#### — 设实体集 $E$ 具有多值属性， $S$ 是 $E$ 对应的关系

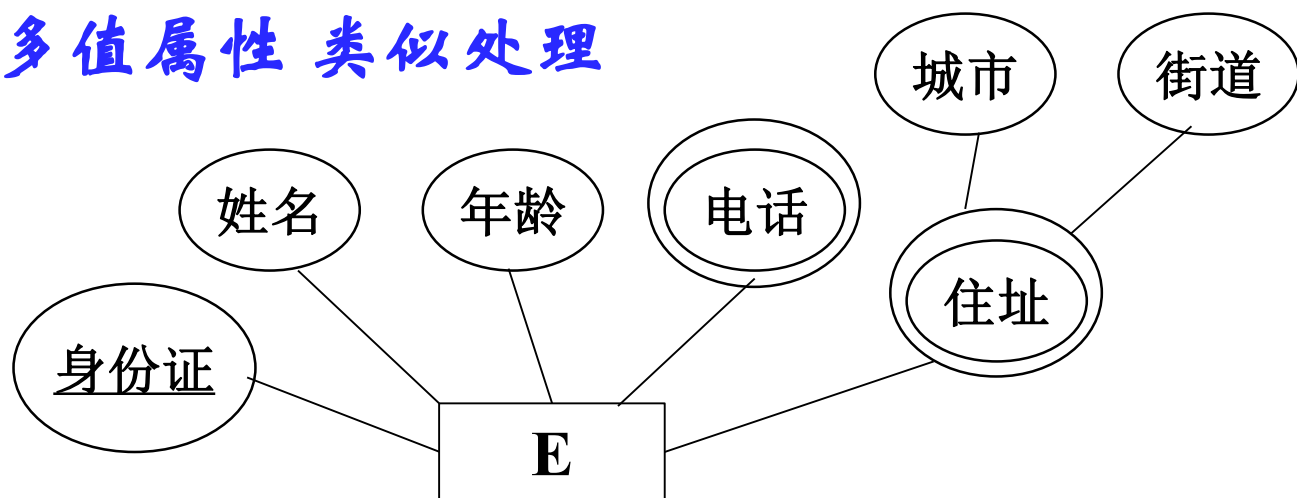
- 为 $E$ 的每个多值属性 $A$ 建立一个关系 $T$ ，用 $T$ 表示 $A$ 。
- 如果 $A$ 是简单属性， $T$ 的属性为 $A$ 与 $S$ 的主键 $K$ 。 $A$ 和 $K$ 形成 $T$ 的主键。
- 如果 $A$ 是复合属性， $T$ 包含 $A$ 的简单子属性和 $S$ 的键 $K$ 。 $A$ 的简单子属性和 $K$ 形成 $T$ 的键。
- $S$ 关系中忽略属性 $A$ 。

#### — 对联系 $R$ 的多值属性类似处理

$S(\underline{\text{身份证}}, \text{姓名}, \text{年龄})$

$T_1(\underline{\text{身份证}}, \underline{\text{城市}}, \underline{\text{街道}})$

$T_2(\underline{\text{身份证}}, \underline{\text{电话}})$



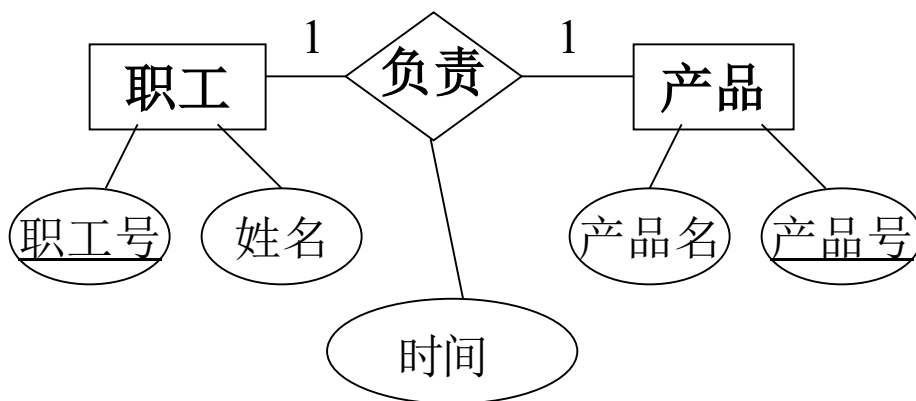


## 4.3.1 形成初始关系数据库模式

### • 实体间联系的变换

#### – 1:1联系的变换

- 设R是实体集 $E_1$ 和 $E_2$ 之间的1:1联系，S和T是 $E_1$ 和 $E_2$ 对应的关系
- **方法1：**通过在S或T中增加有关信息来表示联系R
  - T(或S)的主键作为外部键添入S(或T)；
  - R的简单属性和复合属性的简单子属性作为简单属性添入S(或T)。





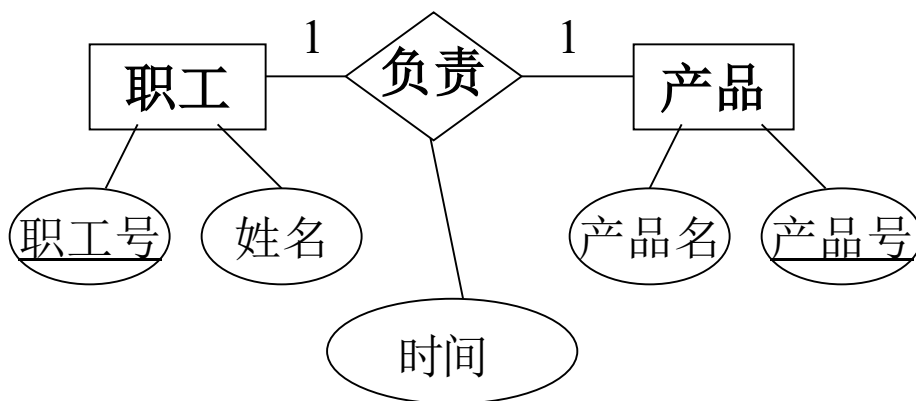


## 4.3.1 形成初始关系数据库模式

### • 实体间联系的变换

#### – 1:1联系的变换

- 设R是实体集 $E_1$ 和 $E_2$ 之间的1:1联系，S和T是 $E_1$ 和 $E_2$ 对应的关系
- 方法2：建立一个单独的关系W表示R
  - T和S的主键作为键添入W；
  - R的简单属性和复合属性的简单子属性作为简单属性添入W。



为降低查询代价  
应尽量减少  
连接操作！

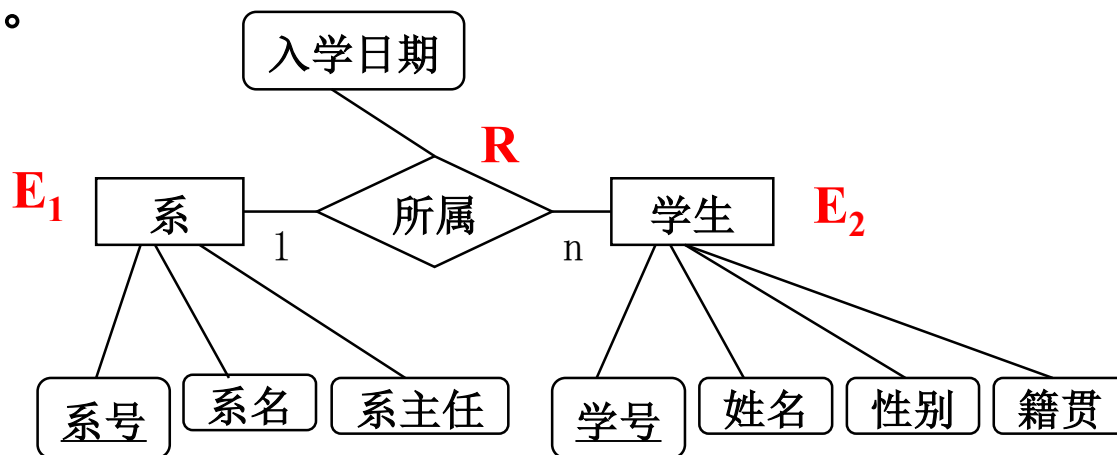


# 4.3.1 形成初始关系数据库模式

## • 实体间联系的变换

### — 1:n联系的变换

- 设R是从实体集 $E_1$ 到实体集 $E_2$ 的1:N联系，S和T是 $E_1$ 和 $E_2$ 对应的关系。
- **方法1**：不需建立新关系。由于T的每个实体至多与S的一个实体对应，因此用T来表示R
  - S的主键作为外部键添入T；
  - R的简单属性和复合属性的简单子属性作为简单属性添入T(或S)。



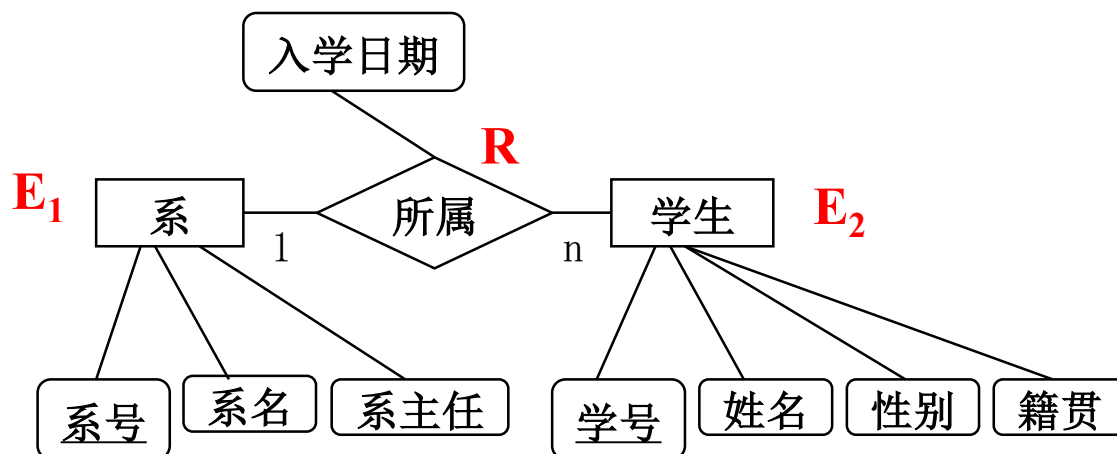


## 4.3.1 形成初始关系数据库模式

- 实体间联系的变换

- 1:n联系的变换

- 设R是从实体集 $E_1$ 到实体集 $E_2$ 的1:N联系，S和T是 $E_1$ 和 $E_2$ 对应的关系。
- 方法2：建立一个单独的关系W表示R，同1:1联系。



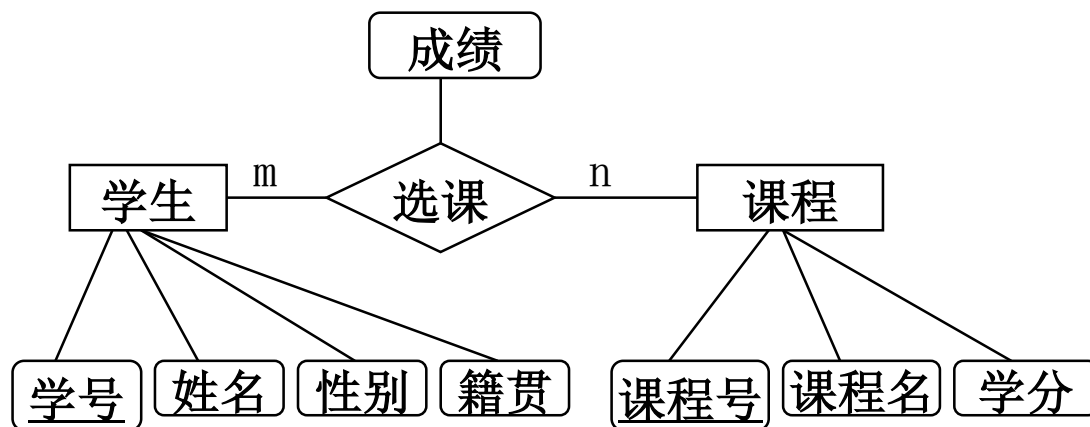


# 4.3.1 形成初始关系数据库模式

## • 实体间联系的变换

### — m:n联系的变换

- 设R是从实体集 $E_1$ 到实体集 $E_2$ 的M:N联系，S和T是 $E_1$ 和 $E_2$ 对应的关系。
- 建立一个新关系W来表示R。
- S和T的主键添入W，既作为外部键，也组合起来作为W的主键。
- W还需要包含R的简单属性和复合属性的简单子属性。





## 4.3.1 形成初始关系数据库模式

- 实体间联系的变换

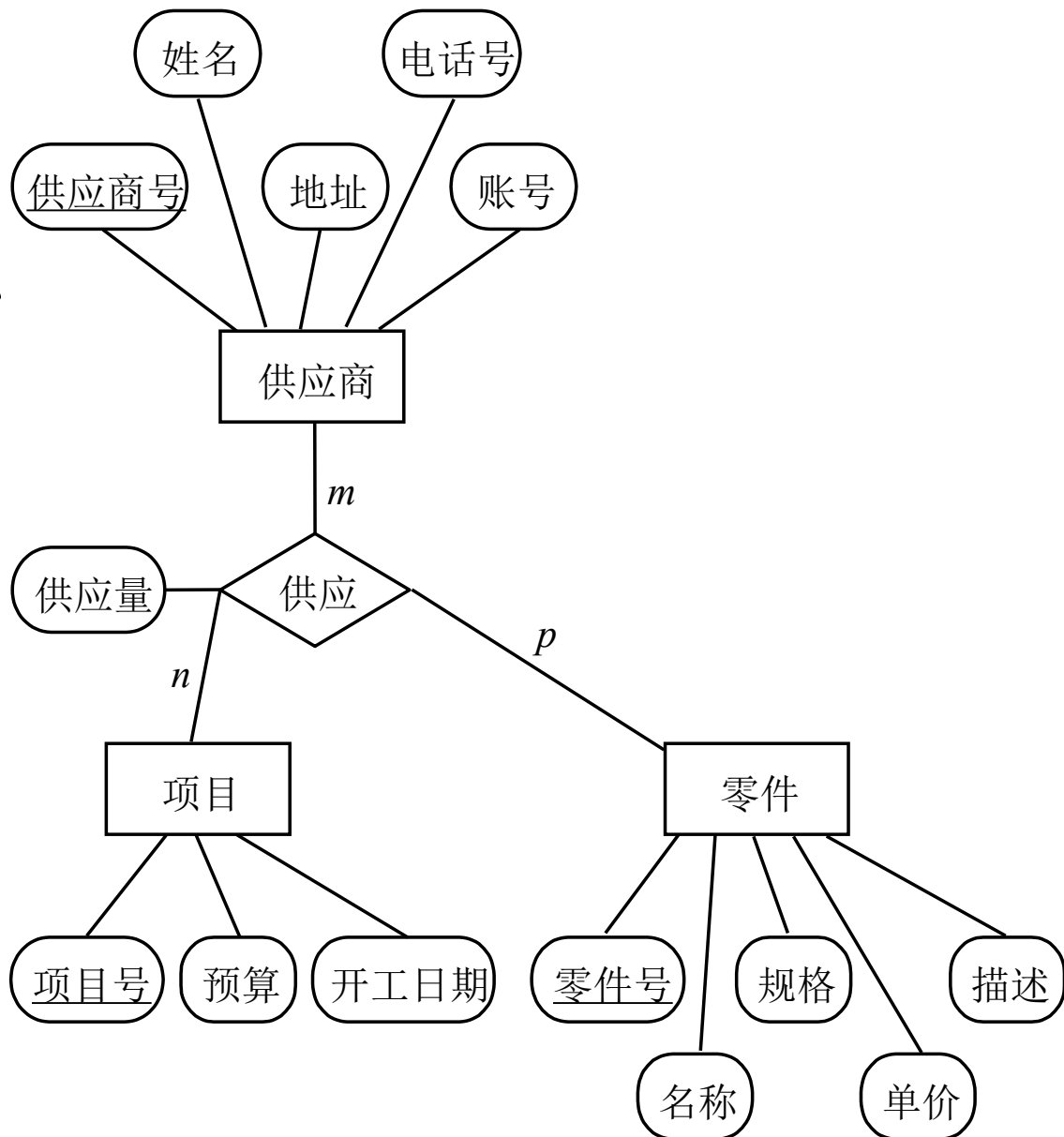
- **n元联系的变换**

- 设R是关联实体集 $E_1$ 、 $E_2$ 、...、 $E_n$ 的n元联系。
    - 类似于M:N联系的表示方法：
      - 需建立一个关系T，用T来表示R。
      - 所有 $E_i$ 的主键都是T的外部键，也组合起来作为T的主键。
      - T还包含R的简单属性和复合属性的简单子属性。





- 实体间联系的变换
  - n元联系的变换
  - 例





## 4.3.1 形成初始关系数据库模式

- 通过前面的步骤，初始关系数据库模式已经形成
- 接下来，对初始关系数据库模式中的每个关系模式进行深入地分析，与用户协商：
  - 确定每个初始关系的函数依赖集
  - 使用关系数据库设计理论，对关系模式进行规范化处理





# 逻辑数据库设计

## • 逻辑数据库设计的步骤

- 形成初始关系数据库模式;
- 函数依赖与关系模式规范化;
- 关系模式优化;
- 定义关系上的完整性和安全性约束;
- 子模式定义;
- 性能估计。







## 4.3.2 关系数据库设计理论

### • 问题的提出

— 初始关系模式不是逻辑设计的最终结果，其中某些关系模式可能存在由属性间的函数依赖引起的问题：

- 冗余问题
- 插入问题
- 更新问题
- 删除问题





## • 问题的提出

— 例如，建立一个描述学校的数据库。

涉及的对象包括：

学生的学号 (Sno)，所在系 (SD)，  
系主任姓名 (MN)，课程名 (CN)，  
成绩 (G)

— 假设学校的数据库模式由一个单一的关系模式 Student 构成，则该关系模式的属性集合为：

$$U = \{Sno, SD, MN, CN, G\}$$





## 现实世界的已知事实:

- 一个系有若干学生, 一个学生只属于一个系
- 一个系只有一个系主任
- 一个学生可以选修多门课, 每门课有若干学生选修
- 每个学生所学的每门课程都有一个成绩

## 关系模式Student(Sno, SD, MN, CN, G)

### 存在的问题:

- 如果一个系刚刚成立, 尚无学生, 我们无法把这个系及其主任的信息存入数据库, 称为**插入异常**。
- 反过来, 如果某个系的学生全部毕业了, 我们在删除该系学生信息的同时, 把这个系及其主任的信息也删掉了, 这称为**删除异常**。





## 现实世界的已知事实：

- 一个系有若干学生，一个学生只属于一个系
- 一个系只有一个系主任
- 一个学生可以选修多门课，每门课有若干学生选修
- 每个学生所学的每门课程都有一个成绩

## 关系模式Student(Sno, SD, MN, CN, G)

### 存在的问题：

- 每个系主任的名字在关系中重复出现，出现次数与该系学生人数相同，称为**数据冗余**。
- 若某个系的系主任更换了，需要修改与该系学生有关的每个元组，称为**更新问题**。





**结论：** Student(Sno, SD, MN, CN, G) **不是一个好的关系模式**

- 一个“好”的模式应当不会发生插入异常、删除异常
- 更新问题、数据冗余应尽可能少。

1. 怎样评价一个关系模式的优劣？
2. 怎样将一个不太好的关系模式分解为一组较理想的关系模式？





# 相关概念

## • 函数依赖

### — 定义1:

- 设 $R$ 是一个关系模式， $U$ 是 $R$ 的属性集合， $X$ 和 $Y$ 是 $U$ 的子集。对于 $R$ 的任意实例 $r$ ， $r$ 中任意两个元组 $t_1$ 和 $t_2$ ，如果 $t_1[X]=t_2[X]$ ，则 $t_1[Y]=t_2[Y]$ ，我们称 $X$ 函数地确定 $Y$ ，或 $Y$ 函数依赖于 $X$ ，记作 $X \rightarrow Y$ 。

只能根据数据的语义来确定函数依赖

例如，“身份证” $\rightarrow$ “年龄”

例如，“姓名” $\rightarrow$ “年龄”这个函数依赖只有在没有同名人的条件下成立





## • 函数依赖

- 如果  $X \rightarrow Y$  而且  $Y$  不是  $X$  的子集，则称  $X \rightarrow Y$  是 **非平凡函数依赖**。若不特别声明，我们总是讨论非平凡函数依赖。
- 如果  $X \rightarrow Y$ ，我们称  $X$  为这个函数依赖的 **决定属性集**。

例：在关系  $SC(Sno, Cno, Grade)$  中，  
非平凡函数依赖：  $(Sno, Cno) \rightarrow Grade$   
平凡函数依赖：  $(Sno, Cno) \rightarrow Sno$   
 $(Sno, Cno) \rightarrow Cno$





## • 函数依赖

### — 定义2:

- 设 $R$ 是一个具有属性集合 $U$ 的关系模式，如果 $X \rightarrow Y$ ，并且对于 $X$ 的任何一个真子集 $Z$ ， $Z \rightarrow Y$ 都不成立，则称 $Y$ 完全函数依赖于 $X$ 。若 $X \rightarrow Y$ ，但 $Y$ 不完全函数依赖于 $X$ ，则称 $Y$ 部分函数依赖于 $X$ 。

### — 定义3:

- 设 $R$ 是一个具有属性集合 $U$ 的关系模式， $X \subseteq U, Y \subseteq U, Z \subseteq U$ ， $Y \rightarrow X$ 不成立， $Z-X$ 、 $Z-Y$ 和 $Y-X$ 不空。如果 $X \rightarrow Y$ ， $Y \rightarrow Z$ ，则称 $Z$ 传递地函数依赖于 $X$ 。







- 例

**Student(Sno, Sname, Ssex, Sage, Sdept)**

**$Sno \rightarrow Sname, Sno \rightarrow Ssex, Sno \rightarrow Sage, Sno \rightarrow Sdept$**

**$(Sno, Sname) \rightarrow Sdept,$**

**$(Sno, Ssex) \rightarrow Sdept$**

**例：在关系Std(Sno, Sdept, Mname)中，有：**

**$Sno \rightarrow Sdept, Sdept \rightarrow Mname,$**

**Mname传递函数依赖于Sno。**





# 函数依赖的公理系统

- 函数依赖

- 定义6:

- 设 $R$ 是一个具有属性集合 $U$ 的关系模式,  $F$ 是 $R$ 上的函数依赖集合。如果对于 $R$ 的任意一个使 $F$ 成立的关系实例 $r$ , 函数依赖 $X \rightarrow Y$ 都成立, 则称 $F$ 逻辑地蕴含 $X \rightarrow Y$ 。

- 例:

- 给定关系模式 $R(A, B, C, G, H, I)$ 及函数依赖 $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$ , 则

函数依赖 $A \rightarrow H$ 被 $F$ 逻辑蕴含





# 函数依赖的推理系统

- 函数依赖的推理系统

- 在关系模式的规范化处理过程中，只知道一个给定的函数依赖集合是不够的。还需要知道由给定的函数依赖集合所蕴涵的所有函数依赖的集合。为了能够从给定的函数依赖集合推导出这个集合蕴涵的所有函数依赖，我们需要一个有效而完备的推理系统。
- Armstrong公理就是这样一个系统。





# 函数依赖的推理系统

- **Armstrong公理**

- 一套推理规则，是模式分解算法的理论基础

- 用途

- 求给定关系模式的候选键

- 从一组函数依赖求得蕴含的函数依赖





# 函数依赖的推理系统

## • Armstrong公理

— 包含如下三条推理规则：

- A1. 自反律 (Reflexivity): 若  $Y \subseteq X \subseteq U$ , 则  $X \rightarrow Y$  为  $F$  所蕴含。
- A2. 增广律 (Augmentation): 若  $X \rightarrow Y$  为  $F$  所蕴含, 且  $Z \subseteq U$ , 则  $XZ \rightarrow YZ$  为  $F$  所蕴含。
- A3. 传递律 (Transitivity): 若  $X \rightarrow Y$  及  $Y \rightarrow Z$  为  $F$  所蕴含, 则  $X \rightarrow Z$  为  $F$  所蕴含。

注意：由自反律所得到的函数依赖均是平凡的函数依赖。

已被证明是正确有效的、且完备的





# 函数依赖的推理系统

## • 导出规则

1. 根据A1, A2, A3这三条推理规则可以得到下面三条推理规则:

— 合并规则: 由 $X \rightarrow Y$ ,  $X \rightarrow Z$ , 有 $X \rightarrow YZ$ 。

(A2, A3)  $X \rightarrow YX$ ,  $XY \rightarrow ZY$

— 伪传递规则: 由 $X \rightarrow Y$ ,  $WY \rightarrow Z$ , 有 $XW \rightarrow Z$ 。

(A2, A3)  $XW \rightarrow YW$

— 分解规则: 由 $X \rightarrow Y$ 及 $Z \subseteq Y$ , 有 $X \rightarrow Z$ 。

(A1, A3)  $Z \subseteq Y$ ,  $Y \rightarrow Z$





# 函数依赖的推理系统

- 导出规则

2. 根据合并规则和分解规则，可得引理6.1

引理6.1  $X \rightarrow A_1 A_2 \dots A_k$  成立的充分必要条件是  $X \rightarrow A_i$  成立 ( $i=1, 2, \dots, k$ )。





# 函数依赖的推理系统

## • 闭包

### — 定义7:

- 在关系模式  $R<U, F>$  中为  $F$  所逻辑蕴含的函数依赖的全体叫作  $F$  的闭包，记为  $F^+$ 。

— 例如：给定关系模式  $R=(A,B,C,G,H,I)$  及函数依赖  $F=\{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$ , 则

»  $A \rightarrow H, CG \rightarrow HI, AG \rightarrow I, \dots$

- 设  $F$  为属性集  $U$  上的一组函数依赖， $X \subseteq U$ ， $X_F^+ = \{A | X \rightarrow A \text{ 能由 } F \text{ 根据 Armstrong 公理导出}\}$ ， $X_F^+$  称为属性集  $X$  关于函数依赖集  $F$  的闭包。







# 函数依赖的推理系统

— 例：关系模式  $R(A_1, A_2, A_3)$  的函数依赖集  $F$  为  $\{A_1 \rightarrow A_2, A_2 \rightarrow A_3\}$

- 若  $X=A_1, X^+ = \{A_1, A_2, A_3\}$
- 若  $X=A_2, X^+ = \{A_2, A_3\}$
- 若  $X=A_3, X^+ = \{A_3\}$





# 函数依赖的推理系统

- 计算闭包：求属性集 $X$  ( $X \subseteq U$ ) 关于 $U$ 上的函数依赖集 $F$ 的闭包 $X_F^+$ 
  - 令 $X^{(0)} = X$ ,  $i=0$
  - 求 $B$ , 这里 $B = \{ A \mid (\exists V)(\exists W)(V \rightarrow W \in F \wedge V \subseteq X^{(i)} \wedge A \in W) \}$ ;  
对 $X^{(i)}$ 中的每个元素, 依次检查相应的函数依赖, 将依赖它的属性加入 $B$
  - $X^{(i+1)} = B \cup X^{(i)}$
  - 判断 $X^{(i+1)} = X^{(i)}$  吗?
  - 若相等或 $X^{(i)} = U$ , 则 $X^{(i)}$  就是 $X_F^+$ , 算法终止。
  - 若否, 则 $i=i+1$ , 返回第 (2) 步。

算法至多执行 $|U-X|$ 次循环



# 函数依赖的推理系统

## • 例

- $U = \{A, B, C, D, E\}$ ,
- $F = \{AB \rightarrow C, AC \rightarrow B, B \rightarrow D, C \rightarrow E, EC \rightarrow B\}$ ,
- 求  $(AB)^+$ 。

解 设  $X^{(0)} = \{AB\}$ ;

(1) 计算  $X^{(1)}$ : 逐一的扫描  $F$  集合中各个函数依赖, 找左部为  $A$ ,  $B$  或  $AB$  的函数依赖。得到两个:  $AB \rightarrow C$ ,  $B \rightarrow D$ 。

于是  $X^{(1)} = \{AB\} \cup \{CD\} = \{ABCD\}$ 。





# 函数依赖的推理系统

(2) 因为  $X^{(0)} \neq X^{(1)}$ ，所以再找出左部为  $\{ABCD\}$  子集的那些函数依赖，又得到  $AB \rightarrow C$ ,  $B \rightarrow D$ ,  $C \rightarrow E$ ,  $AC \rightarrow B$ ,

于是  $X^{(2)} = X^{(1)} \cup \{BCDE\} = \{ABCDE\}$ 。

(3) 因为  $X^{(2)} = U$ ，算法终止

所以  $(AB)_F^+ = \{ABCDE\}$ 。意味着什么？

$(AB)_F^+ = \{A, B, C, D, E\} = U$ ,  
因此，AB是该关系的候选键！





例. 设  $R\langle U, F \rangle$ , 其中  $U = \{A, B, C, D, E, I\}$ ;

$F = \{A \rightarrow D, AB \rightarrow E, BI \rightarrow E, CD \rightarrow I, E \rightarrow C\}$

求:  $(AE)_F^+$

解: 令  $X^{(0)} = \{AE\}$ , 在  $F$  中找出左边是  $AE$  子集的函数依赖, 其结果是:  $A \rightarrow D, E \rightarrow C$ , 所以  $X^{(1)} = X^{(0)} \cup \{CD\} = \{ACDE\}$ ;

$X^{(0)} \neq X^{(1)}$  在  $F$  中找出左边是  $\{AEDC\}$  子集的函数依赖, 其结果是  $CD \rightarrow I$ , 所以  $X^{(2)} = X^{(1)} \cup \{I\} = \{ACDEI\}$ ;

显然  $X^{(2)} \neq X^{(1)}$ ,

但  $F$  中未用过的函数依赖的左边属性已没有  $X^{(2)}$  的子集, 所以不必再计算下去。即  $(AE)_F^+ = \{ACDEI\}$ 。

关系的候选键?





## • 快速求解候选键的方法

对于给定的关系模式 $R(A_1, A_2, \dots, A_n)$ 和函数依赖集 $F$ , 可将其属性分为四类:

- L类: 仅出现在 $F$ 的函数依赖左部的属性
- R类: 仅出现在 $F$ 的函数依赖右部的属性
- N类: 在 $F$ 的函数依赖左右两边均未出现的属性
- LR类: 在 $F$ 的函数依赖左右两边均出现的属性





**定理1:** 对于给定的关系模式及其函数依赖集F, 若  
 $X(X \in R)$  是L类属性, 则X必为R的任一候选键  
的成员

**例:** 设  $R(A,B,C,D), F=\{D \rightarrow B, B \rightarrow D, AD \rightarrow B, AC \rightarrow D\}$   
求R的候选键

**解:** 考察F发现, A,C两属性是L类属性, AC必为  
R的候选键成员, 又因为  $(AC)_F^+ = \{ACBD\}$ .

所以, AC是候选键。





**定理2：对于给定的关系模式R及其F，如果X( $X \in R$ )是R类属性，则X不在任何候选键中。**

**定理3：设有R及其F，如果X是R的N类属性，则X必包含在R的任一候选键中。**

**推论：对于给定的关系模式R及其函数依赖集F，如果X是R的N类和L类组成的属性集，且 $X_F^+$ 包含了R的全部属性，则X是R的唯一候选键。**







**例： 设 $R(A,B,C,D,E,P)$ ,**

**$F=\{A \rightarrow D, E \rightarrow D, D \rightarrow B, BC \rightarrow D, DC \rightarrow A\}$ ,**

**求 $R$ 的候选键。**

**解：  $C, E$ 是 $L$ 类属性，  $P$ 是 $N$ 类属性。**

**$(CEP)_F^+ = ABCDEP$ 。 所以 $CEP$ 是候选键。**





## — 定义8：极小函数依赖集

- 如果函数依赖集F满足下列条件，则称F为一个极小函数依赖集。亦称为最小依赖集或最小覆盖：
  - F中任一函数依赖的右部仅含有一个属性
  - F中不存在这样的函数依赖 $X \rightarrow A$ ，使得F与 $F - \{X \rightarrow A\}$ 等价
  - F中不存在这样的函数依赖 $X \rightarrow A$ ，X有真子集Z使得 $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ 与F等价

## — 极小函数依赖集不唯一

- $F = \{A \rightarrow B, B \rightarrow A, B \rightarrow C, A \rightarrow C, C \rightarrow A\}$ 的极小函数依赖集为
  - $F1 = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$ ，或
  - $F2 = \{A \rightarrow B, B \rightarrow A, A \rightarrow C, C \rightarrow A\}$





对于关系模式  $U=\{SNO, SDEPT, MN, CNAME, G\}$ , 其中,  
学生的学号 (SNO), 所在系 (SDEPT), 系主任姓名 (MN),  
课程名 (CNAME), 成绩 (G)

(1)  $F=\{SNO \rightarrow SDEPT, SDEPT \rightarrow MN, (SNO, CNAME) \rightarrow G\}$

$F$ 是最小覆盖么?

是!

(2)  $F'=\{SNO \rightarrow SDEPT, SNO \rightarrow MN, SDEPT \rightarrow MN,$   
 $(SNO, CNAME) \rightarrow G, (SNO, SDEPT) \rightarrow SDEPT\}$

$F'$ 是最小覆盖么?

不是! 因为:  $F' - \{SNO \rightarrow MN\}$  与  $F'$  等价

$F' - \{(SNO, SDEPT) \rightarrow SDEPT\}$  也与  $F'$  等价





- 给定一个函数依赖集 $F$ ，如何计算 $F$ 极小函数依赖集

根据定义：

- 逐一检查 $F$ 中各函数依赖 $X \rightarrow Y$ ，若 $Y = A_1 A_2 \dots A_k$ ， $k \geq 2$ ，则用 $\{X \rightarrow A_j | j=1, 2, \dots, k\}$ 来取代 $X \rightarrow Y$
- 循环地检查 $F$ 中各函数依赖 $X \rightarrow A$ ，令 $G = F - \{X \rightarrow A\}$ ，若 $A \in X_G^+$ ，则从 $F$ 中去掉此函数依赖
- 循环地取出 $F$ 中各函数依赖 $X \rightarrow A$ ，设 $X = B_1 B_2 \dots B_m$ ，逐一考查 $B_i$  ( $i=1, 2, \dots, m$ )，若 $A \in (X - B_i)_F^+$ ，则以 $X - B_i \rightarrow A$ 取代 $X \rightarrow A$ 。





# 关系模式的规范形式

- 范式是符合某一种级别要求的关系模式的集合。
- 关系数据库中的关系必须满足一定的要求。满足不同程度要求的为不同范式
- 关系模式的范式主要有四种
  - 第一范式 (1NF)
  - 第二范式 (2NF)
  - 第三范式 (3NF)
  - BC范式 (BCNF)
  - 多值依赖与第四范式(4NF)

满足这些范式条件的关系模式可以在不同程度上避免“冗余问题、插入问题、更新问题和删除问题”



# 关系模式的规范形式

- 各种范式之间存在联系

$$1NF \supset 2NF \supset 3NF \supset BCNF \supset 4NF \supset 5NF$$

- 某一关系模式R为第n范式，可简记为  
 $R \in nNF$





# 关系模式的规范形式

- 第一范式(1NF)

- 定义:

- 设 $R$ 是一个关系模式。如果 $R$ 的每个属性的值域都是不可分的简单数据项的集合，则称这个关系模式为第一范式关系模式，记作1NF。

- 在任何一个关系数据库系统中，第一范式都是一个最起码的要求。在以后的讨论中，我们假定所有关系模式都是1NF。





- 例，关系模式  $SLC(Sno, Sdept, Sloc, Cno, Grade)$ ,  
Sloc为学生住处，假设每个系的学生住在同一个地方
- 函数依赖包括：

$(Sno, Cno) \rightarrow Grade$

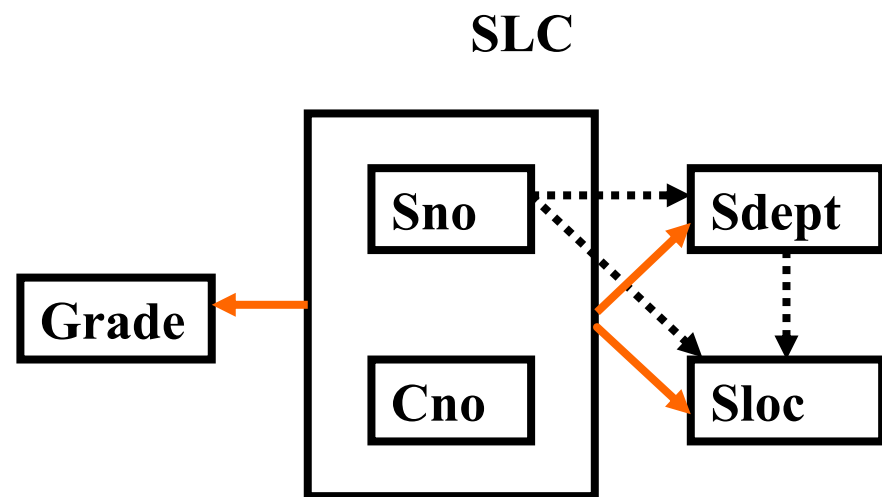
$Sno \rightarrow Sdept$

$(Sno, Cno) \rightarrow Sdept$

$Sno \rightarrow Sloc$

$(Sno, Cno) \rightarrow Sloc$

$Sdept \rightarrow Sloc$



SLC的候选键为(Sno, Cno)

SLC满足第一范式

非键属性Sdept和Sloc部分函数依赖于候选键(Sno, Cno)



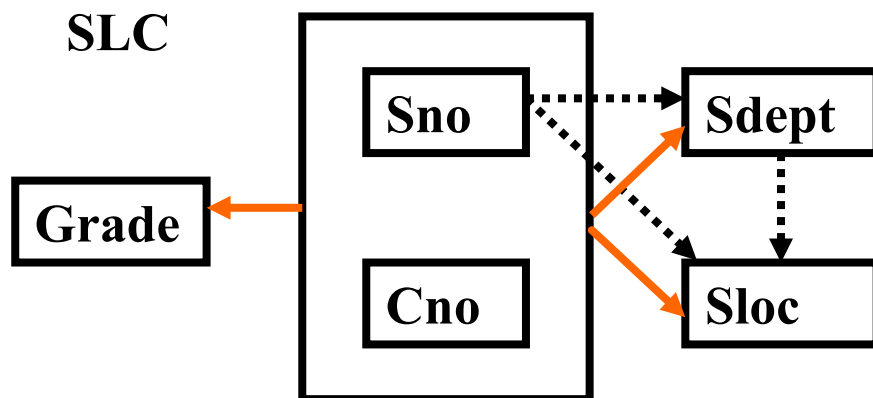




## • SLC存在的问题

- 插入异常
- 删除异常
- 数据冗余度大
- 修改复杂

- 例如学生转系，在修改此学生元组的Sdept值的同时，还可能需修改住处（Sloc）。如果这个学生选修了K门课，则必须无遗漏地修改K个元组中全部Sdept、Sloc信息。

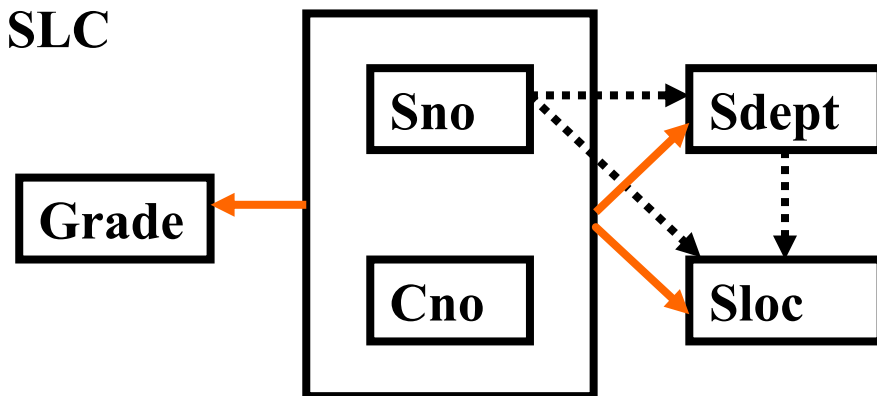


**因此SLC不是一个好的关系模式！**





SLC



- 问题产生原因

- Sdept、Sloc部分函数依赖于候选键(Sno, Cno)

- 解决方法

- 把关系模式SLC(Sno, Sdept, Sloc, Cno, Grade)分解为两个关系模式，以消除这些部分函数依赖：

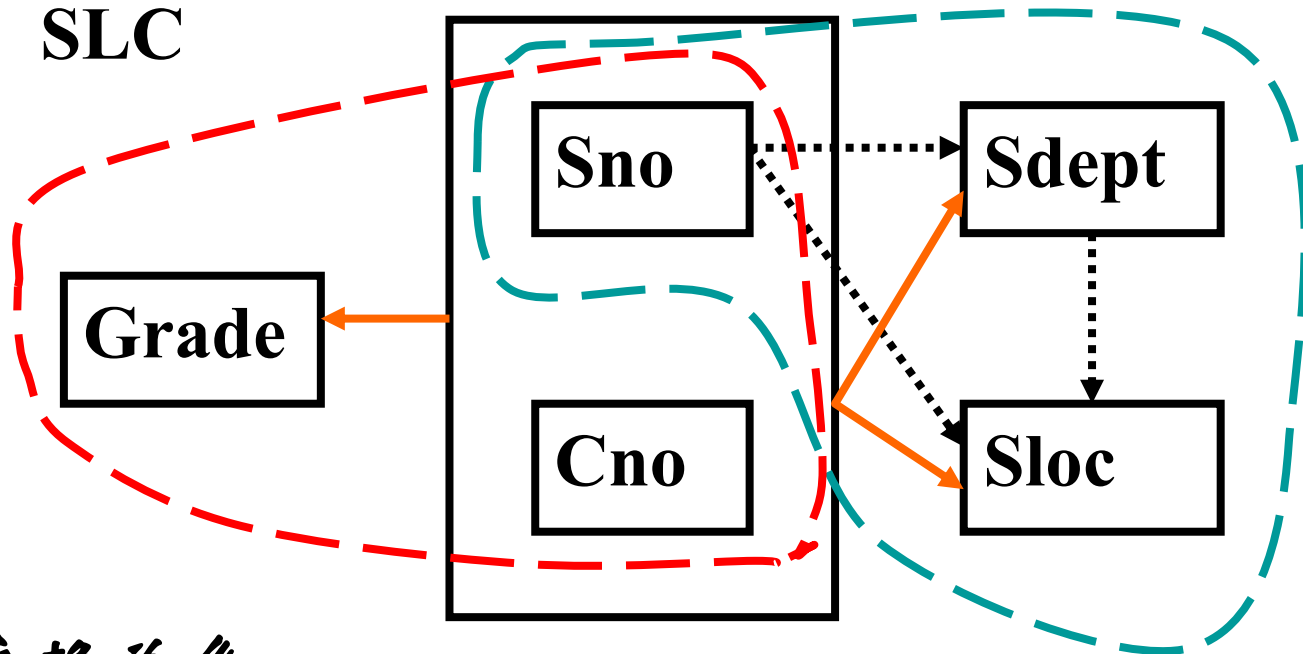
SC (Sno, Cno, Grade)

SL (Sno, Sdept, Sloc)



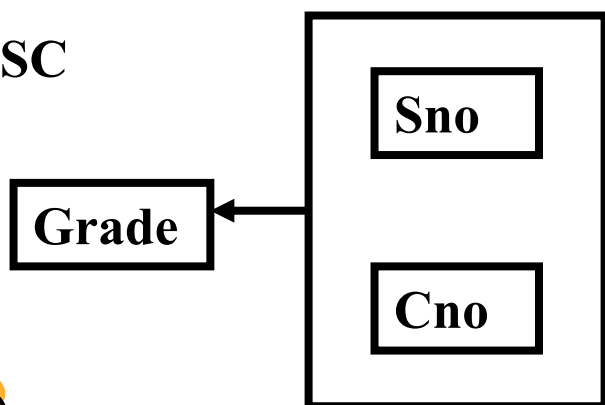


SLC

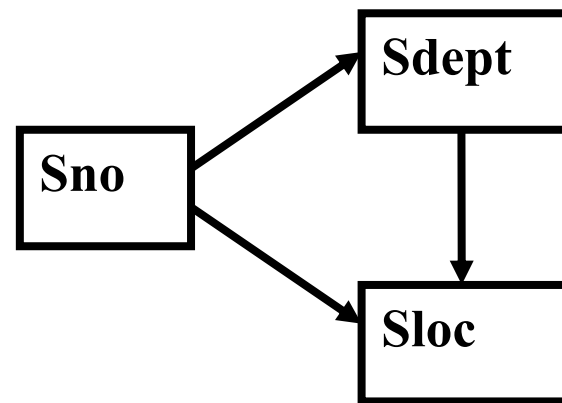


函数依赖关系

SC



SL



使上述四个问题在一定程度上得到了解决



## • 第二范式(2NF)

### — 定义

- 若关系模式R是1NF，而且每一个非键属性都完全函数依赖于R的候选键(码)，则R称为第二范式关系模式，记作**2NF**。

### — 例如，

$SLC(Sno, Sdept, Sloc, Cno, Grade) \in 1NF$

$SC(Sno, Cno, Grade) \in 2NF$

$SL(Sno, Sdept, Sloc) \in 2NF$





- 采用模式分解法将一个1NF的关系分解为多个2NF的关系，可以在一定程度上减轻原1NF关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。
- 但是，将一个1NF关系分解为多个2NF的关系，并不能完全消除关系模式中的各种异常情况和数据冗余。





- 例，2NF关系模式SL(Sno, Sdept, Sloc)中

函数依赖： $Sno \rightarrow Sdept$ ,  $Sdept \rightarrow Sloc$ ,  $Sno \rightarrow Sloc$

Sloc传递函数依赖于Sno，即SL中存在非键属性对候选键的传递函数依赖。

SL(Sno, Sdept, Sloc)存在的问题

— 插入异常

— 删除异常

所以，SL仍不是一个好的关系模式！

— 数据冗余度大

— 修改复杂

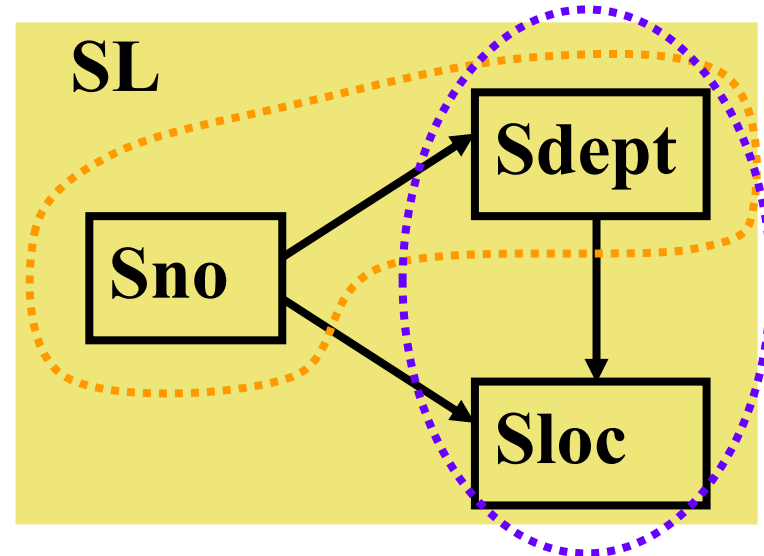
- 当学校调整学生住处时，由于关于每个系的住处信息是重复存储的，修改时必须同时更新该系所有学生的Sloc属性值。





- 问题产生原因

- Sloc传递函数依赖于Sno



- 解决方法

- 把SL分解为两个关系模式，以消除传递函数依赖：

**SD (Sno, Sdept)**

**DL (Sdept, Sloc)**

**SD的码为Sno, DL的码为Sdept**

在分解后的关系模式中既没有非键属性对候选键(码)的部分函数依赖也没有非键属性对候选键(码)的传递函数依赖，在一定程度上解决了上述四个问题！



## • 第三范式(3NF)

### — 定义:

- 如果关系模式R是2NF, 而且它的任何一个非键属性都不传递地依赖于任何候选键, 则R称为第三范式关系模式, 记作**3NF**。

### — 例:

$SL(Sno, Sdept, Sloc) \in 2NF$

$SD(Sno, Sdept) \in 3NF$

$DL(Sdept, Sloc) \in 3NF$

学生(学号, 姓名, 宿舍楼, 宿舍号)  $\in 3NF$







- 如果 $R \in 3NF$ ，则 $R$ 也是 $2NF$
- 若 $R \in 3NF$ ，则 $R$ 的每一个非键属性既不部分函数依赖于候选键也不传递函数依赖于候选键。
- 采用模式分解法将一个 $2NF$ 的关系分解为多个 $3NF$ 的关系，可以在一定程度上解决原 $2NF$ 关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。
- 但是，将一个 $2NF$ 关系分解为多个 $3NF$ 的关系后，**并不能完全消除关系模式中的各种异常情况和数据冗余。**





- 例：在关系模式STJ (S, T, J) 中，S表示学生，T表示教师，J表示课程。

### — 函数依赖：

假设每一教师只教一门课。每门课由若干教师教，但某一学生选定某门课，就确定了一个固定的教师。某个学生选修某个教师的课就确定了所选课的名称。于是有：

$$(S, J) \rightarrow T, (S, T) \rightarrow J, T \rightarrow J$$

(S, J)和(S, T)都可以作为候选键

STJ  $\in$  3NF





## • 关系模式STJ (S, T, J) 存在的问题

### — 插入异常

- 如果某个教师开设了某门课程，但尚未有学生选修，则有关信息也无法存入数据库中。

### — 删除异常

- 如果选修过某门课程的学生全部毕业了，在删除这些学生元组的同时，相应教师开设该门课程的信息也同时丢掉了。





## • 关系模式STJ (S, T, J) 存在的问题

### — 数据冗余度大

- 虽然一个教师只教一门课，但每个选修该教师该门课程的学生元组都要记录这一信息。

### — 修改复杂

- 某个教师开设的某门课程改名后，所有选修了该教师该门课程的学生元组都要进行相应修改。

虽然 $STJ \in 3NF$ ，但它仍不是一个理想的关系模式。





- 问题出现原因

- 键属性J依赖于T，即键属性J部分函数依赖于候选键(S, T)。

- 解决方法

- 将STJ(S,T,J)分解为二个关系模式：

$(S, J) \rightarrow T, (S, T) \rightarrow J, T \rightarrow J$

$SJ(S, J)$

$TJ(T, J)$

SJ的码为 (S, J)，TJ的码为T(每个老师只教1门课)

在分解后的关系模式中没有任何属性对候选键的部分函数依赖和传递函数依赖。它解决了上述四个问题！



## • BC范式(BCNF)

– Boyce和Codd提出的，比3NF更进了一步。通常认为BCNF是修正的第三范式，所以有时也称为**扩展的第三范式**。

– 定义：

• 设关系模式R是1NF。如果对于R的每个函数依赖 $X \rightarrow Y$ ，则X必为候选键，则R是**BCNF**范式。





- 例：

关系模式SJP(S,J,P);

S表示学生，J表示课程，P表示名次。

每个学生每门课程都有一个确定的名次，

每门课程中每一名次只有一个学生。

由这些语义得到下面的函数依赖：

$\{S,J\} \rightarrow P$ 和 $\{J,P\} \rightarrow S$ 。

$\{S,J\}$ 与 $\{J,P\}$ 都是候选键。

这个关系模式中显然没有属性对候选键的传递依赖或部分依赖。

SPJ是3NF，同时也是BCNF。



## • BCNF的关系模式所具有的性质

- 所有非键属性都完全函数依赖于每个候选键；
- 所有键属性都完全函数依赖于每个**不包含它**的候选键；
- 没有任何属性完全函数依赖于非码的任何一组属性。







- 如果关系模式  $R \in BCNF$ ，必定有  $R \in 3NF$
- 如果一个关系数据库中的所有关系模式都属于  $BCNF$ ，那么在函数依赖范畴内，它已实现了模式的彻底分解，达到了**最高的规范化程度**，消除了插入异常和删除异常。





- 关系模式的分类：

- **静态关系**：一旦数据已加载，用户只能在这个关系上运行查询操作，不再进行插入、删除和更新操作。

- **动态关系**：经常被更新、插入和删除。

- 静态关系只需具有第一规范形式

- 动态关系应该具有第三规范形式





- 关系模式规范化：关系模式分解。
  - 把一个关系模式分解为几个子模式，使得这些子模式具有指定的规范化形式。
- 关系模式规范化的基本步骤

**1NF**

↓ 消除非键属性对候选键的部分函数依赖

**2NF**

↓ 消除非键属性对候选键的传递函数依赖

**3NF**

↓ 消除键属性对候选键的部分和传递函数依赖

**BCNF**





- 不能说规范化程度越高的关系模式就越好。
  - 在设计数据库模式结构时，必须对现实世界的实际情况和用户应用需求作进一步分析，确定一个合适的、能够反映现实世界的模式。
  - 这也就是说，上面的规范化步骤可以在其中任何一步终止。





- 将一个关系模式  $R\langle U, F \rangle$  分解为若干个关系模式  $R_1\langle U_1, F_1 \rangle$ ,  $R_2\langle U_2, F_2 \rangle$ , ...,  $R_n\langle U_n, F_n \rangle$  (其中  $U = U_1 \cup U_2 \cup \dots \cup U_n$ , 且不存在  $U_i \subseteq U_j$ ,  $F_i$  为  $F$  在  $U_i$  上的投影)
  - 例如, 将  $STJ(S, T, J)$  分解为  $ST(S, T)$  和  $SJ(S, J)$
- 关系模式的分解意味着相应地将存储在一个二维表  $T$  中的数据分散到若干个二维表  $T_1, T_2, \dots, T_n$  中去 (其中  $T_i$  是  $T$  在属性集  $U_i$  上的投影)。



- 例，关系模式R的属性集合 $U=\{S\#,SD,MN\}$ ，函数依赖集合 $F=\{S\#\rightarrow SD,SD\rightarrow MN\}$ 。
- R的关系实例：

S#	SD	MN
S1	D1	张红
S2	D1	张红
S3	D2	李微
S4	D3	王力

- 该模式存在的问题？





- 例，关系模式R的属性集合 $U=\{S\#,SD,MN\}$ ，函数依赖集合 $F=\{S\#\rightarrow SD, SD\rightarrow MN\}$ 。

— 分解1:  $\rho_1 = \{R_1(S\#), R_2(SD), R_3(MN)\}$

$R_1$	$R_2$	$R_3$
S1	D1	张红
S2	D2	李薇
S3	D3	王力
S4		

如果分解后的关系可以通过自然连接恢复为原来的关系，那么这种分解就没有丢失信息。

分解后的数据库丢失了许多信息，例如无法查询95001学生所在系或该系主任名字。因此这种分解方法是不可取的。





- 例，关系模式R的属性集合 $U=\{S\#,SD,MN\}$ ，函数依赖集合 $F=\{S\#\rightarrow SD, SD\rightarrow MN\}$ 。

— 分解2:  $\rho_2 = \{R_1(S\#, SD), R_2(S\#, MN)\}$

$R_1$		$R_2$	
S1	D1	S1	张红
S2	D1	S2	张红
S3	D2	S3	李薇
S4	D3	S4	王力

虽然分解后的关系可以通过自然连接恢复为原来的关系，但是函数依赖 $SD\rightarrow MN$ 丢失了。







- 例，关系模式R的属性集合 $U=\{S\#,SD,MN\}$ ，函数依赖集合 $F=\{S\#\rightarrow SD, SD\rightarrow MN\}$ 。
  - 分解3:  $\rho_3 = \{R_1(S\#,SD), R_2(SD,MN)\}$

R1		R2	
S1	D1	D1	张红
S2	D1	D2	李薇
S3	D2	D3	王力
S4	D3		

分解后既可以通过自然连接恢复为原来的关系，同时又保证了函数依赖关系不丢失。





- 关系模式分解必须满足：

- 分解的无损连接性
- 函数依赖保持性





- 具有无损连接性的模式分解

- 设关系模式  $R\langle U, F \rangle$  被分解为若干个关系模式  $R_1\langle U_1, F_1 \rangle$ ,  $R_2\langle U_2, F_2 \rangle$ , ...,  $R_n\langle U_n, F_n \rangle$

- 其中:  $U = U_1 \cup U_2 \cup \dots \cup U_n$ , 且不存在  $U_i \subseteq U_j$ ,  $F_i$  为  $F$  在  $U_i$  上的投影

- 若  $R$  与  $R_1, R_2, \dots, R_n$  自然连接的结果相等, 则称关系模式  $R$  的这个分解具有无损连接性 (Lossless join)

- 只有具有无损连接性的分解才能够保证不丢失信息





- 判断对关系模式的一个分解是否与原关系模式等价的标准

(1) 具有无损连接性

(2) 保持函数依赖

(3) 既具有无损连接性，又保持函数依赖



• 如何判断一个分解的无损连接性

[例]:  $U=\{A,B,C,D,E\}$ ,

$F=\{A\rightarrow C, B\rightarrow C, C\rightarrow D, DE\rightarrow C, CE\rightarrow A\}$

$\rho=\{(A, D), (A, B), (B, E), (C, D, E), (A, E)\}$

初始化矩阵

	A	B	C	D	E
AD	$a_1$	$b_{12}$	$b_{13}$	$a_4$	$b_{15}$
AB	$a_1$	$a_2$	$b_{23}$	$b_{24}$	$b_{25}$
BE	$b_{31}$	$a_2$	$b_{33}$	$b_{34}$	$a_5$
CDE	$b_{41}$	$b_{42}$	$a_3$	$a_4$	$a_5$
AE	$a_1$	$b_{52}$	$b_{53}$	$b_{54}$	$a_5$

$A\rightarrow C$

	A	B	C	D	E
AD	$a_1$	$b_{12}$	$b_{13}$	$a_4$	$b_{15}$
AB	$a_1$	$a_2$	$b_{13}$	$b_{24}$	$b_{25}$
BE	$b_{31}$	$a_2$	$b_{33}$	$b_{34}$	$a_5$
CDE	$b_{41}$	$b_{42}$	$a_3$	$a_4$	$a_5$
AE	$a_1$	$b_{52}$	$b_{13}$	$b_{54}$	$a_5$

每一列对应一个属性,

每一行对应分解中的一个关系模式

若属性 $A_j$ 属于关系模式 $R_i$ 中的 $U_i$ , 则在矩阵的j列i行处填 $a_j$ , 否则填 $b_{ij}$

$B\rightarrow C$

	A	B	C	D	E
AD	$a_1$	$b_{12}$	$b_{13}$	$a_4$	$b_{15}$
AB	$a_1$	$a_2$	$b_{13}$	$b_{24}$	$b_{25}$
BE	$b_{31}$	$a_2$	$b_{13}$	$b_{34}$	$a_5$
CDE	$b_{41}$	$b_{42}$	$a_3$	$a_4$	$a_5$
AE	$a_1$	$b_{52}$	$b_{13}$	$b_{54}$	$a_5$



[例]:  $U=\{A,B,C,D,E\}$ ,

$F=\{A\rightarrow C, B\rightarrow C, C\rightarrow D, DE\rightarrow C, CE\rightarrow A\}$

$\rho=\{(A, D), (A, B), (B, E), (C, D, E), (A, E)\}$

$B\rightarrow C$

	A	B	C	D	E
AD	a <sub>1</sub>	b <sub>12</sub>	b <sub>13</sub>	a <sub>4</sub>	b <sub>15</sub>
AB	a <sub>1</sub>	a <sub>2</sub>	b <sub>13</sub>	b <sub>24</sub>	b <sub>25</sub>
BE	b <sub>31</sub>	a <sub>2</sub>	b <sub>13</sub>	b <sub>34</sub>	a <sub>5</sub>
CDE	b <sub>41</sub>	b <sub>42</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>
AE	a <sub>1</sub>	b <sub>52</sub>	b <sub>13</sub>	b <sub>54</sub>	a <sub>5</sub>

$DE\rightarrow C$

	A	B	C	D	E
AD	a <sub>1</sub>	b <sub>12</sub>	b <sub>13</sub>	a <sub>4</sub>	b <sub>15</sub>
AB	a <sub>1</sub>	a <sub>2</sub>	b <sub>13</sub>	a <sub>4</sub>	b <sub>25</sub>
BE	b <sub>31</sub>	a <sub>2</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>
CDE	b <sub>41</sub>	b <sub>42</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>
AE	a <sub>1</sub>	b <sub>32</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>

$C\rightarrow D$

	A	B	C	D	E
AD	a <sub>1</sub>	b <sub>12</sub>	b <sub>13</sub>	a <sub>4</sub>	b <sub>15</sub>
AB	a <sub>1</sub>	a <sub>2</sub>	b <sub>13</sub>	a <sub>4</sub>	b <sub>25</sub>
BE	b <sub>31</sub>	a <sub>2</sub>	b <sub>13</sub>	a <sub>4</sub>	a <sub>5</sub>
CDE	b <sub>41</sub>	b <sub>42</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>
AE	a <sub>1</sub>	b <sub>52</sub>	b <sub>13</sub>	a <sub>4</sub>	a <sub>5</sub>

$\rho$ 具有无损连接性

$CE\rightarrow A$

	A	B	C	D	E
AD	a <sub>1</sub>	b <sub>12</sub>	b <sub>13</sub>	a <sub>4</sub>	b <sub>15</sub>
AB	a <sub>1</sub>	a <sub>2</sub>	b <sub>13</sub>	a <sub>4</sub>	b <sub>25</sub>
BE	a <sub>1</sub>	a <sub>2</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>
CDE	a <sub>1</sub>	b <sub>42</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>
AE	a <sub>1</sub>	b <sub>32</sub>	a <sub>3</sub>	a <sub>4</sub>	a <sub>5</sub>



例：  $U=\{A,B,C,D,E\}$ ,  $F=\{AB\rightarrow C, C\rightarrow D, D\rightarrow E\}$

$\rho = \{(A, B, C), (C, D), (D, E)\}$

试问  $\rho$  是否具有无损连接性？



[例]:  $U=\{A,B,C,D,E\}$ ,  $F=\{AB\rightarrow C, C\rightarrow D, D\rightarrow E\}$

$\rho = \{(A, B, C), (C, D), (D, E)\}$

$AB\rightarrow C$

	A	B	C	D	E
ABC	$a_1$	$a_2$	$a_3$	$b_{14}$	$b_{15}$
CD	$b_{21}$	$b_{22}$	$a_3$	$a_4$	$b_{25}$
DE	$b_{31}$	$b_{32}$	$b_{33}$	$a_4$	$a_5$

	A	B	C	D	E
ABC	$a_1$	$a_2$	$a_3$	$b_{14}$	$b_{15}$
CD	$b_{21}$	$b_{22}$	$a_3$	$a_4$	$b_{25}$
DE	$b_{31}$	$b_{32}$	$b_{33}$	$a_4$	$a_5$

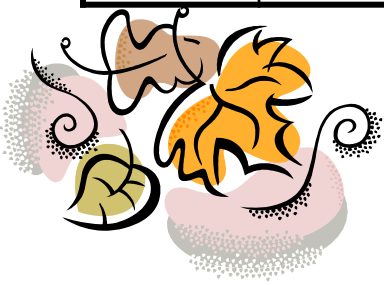
$C\rightarrow D$

	A	B	C	D	E
ABC	$a_1$	$a_2$	$a_3$	$a_4$	$b_{15}$
CD	$b_{21}$	$b_{22}$	$a_3$	$a_4$	$b_{25}$
DE	$b_{31}$	$b_{32}$	$b_{33}$	$a_4$	$a_5$

$D\rightarrow E$

	A	B	C	D	E
ABC	$a_1$	$a_2$	$a_3$	$a_4$	$a_5$
CD	$b_{21}$	$b_{22}$	$a_3$	$a_4$	$a_5$
DE	$b_{31}$	$b_{32}$	$b_{33}$	$a_4$	$a_5$

$\rho$  具有无损连接性







- 当关系模式R被分解为两个子模式时, 下述定理给出了一个判别无损连接性的简单方法

定理: 设 $\rho=\{R_1, R_2\}$ 是关系模式R的一个分解,  
 $U_1$ 、 $U_2$ 和 $U$ 分别是 $R_1$ 、 $R_2$ 和 $R$ 的属性集合,  
 $F$ 是 $R$ 的函数依赖集合。

$\rho$ 具有无损连接性的充分必要条件是:

$U_1 \cap U_2 \rightarrow U_1 - U_2 \in F^+$  或  $U_1 \cap U_2 \rightarrow U_2 - U_1 \in F^+$ 。





[例]  $R=ABC, F=\{A \rightarrow B, B \rightarrow C\},$

$$\rho_1 = \{R_1(AB), R_2(AC)\}$$

$$R_1 \cap R_2 = A, R_1 - R_2 = B$$

由  $A \rightarrow B$  , 得到  $\rho_1$  是无损连接分解

$$\rho_2 = \{R_1(AC), R_2(BC)\}$$

$$R_1 \cap R_2 = C, R_1 - R_2 = A, R_2 - R_1 = B$$

$C \rightarrow A, C \rightarrow B$  均不成立,

所以  $\rho_2$  不是无损连接分解





# 分解的函数依赖保持性

- 函数依赖的投影

- 设  $R(U, F)$  是一个关系模式,  $\rho = \{R_1(U_1, F_1), R_2(U_2, F_2), \dots, R_n(U_n, F_n)\}$  是  $R$  的一个分解.  $F$  在  $R_i$  的属性集合  $U_i$  上的投影是  $F_i = \{X \rightarrow Y \mid X \rightarrow Y \in F^+, X, Y \in U_i\}$ .

例.  $SL(Sno, Sdept, Sloc)$  的函数依赖集合和分解分别为:

$F = \{Sno \rightarrow Sdept, Sdept \rightarrow Sloc, Sno \rightarrow Sloc\}$

$\rho = \{SD(Sno, Sdept), DL(Sdept, Sloc)\}$

$F$  在  $SD$  上的投影为  $F_{SD} = \{Sno \rightarrow Sdept\}$ ;

$F$  在  $DL$  上的投影为  $F_{DL} = \{Sdept \rightarrow Sloc\}$ .





# —分解的函数依赖保持性

- 函数依赖的保持性

- 设 $R(U, F)$ 是关系模式,  $\rho = \{R_1, \dots, R_k\}$ 是 $R$ 的一个分解,  $U_i$ 是 $R_i$ 的属性集,  $F_i$ 是 $F$ 在 $U_i$ 上的投影。如果 $F^+ = \left( \bigcup_{i=1}^K F_i \right)^+$ , 则称 $\rho$ 具有函数依赖保持性。





引理6.3  $F^+ = G^+$  等价 iff  $F \subseteq G^+$  且  $G \subseteq F^+$ .

## • 函数依赖保持性的判别算法

输入:  $F, F_1, F_2, \dots, F_k$ , 记  $G = \bigcup_{i=1}^K F_i$

输出: 是否  $F^+ = G^+$

1. FOR 每个  $X \rightarrow Y \in F$  DO

IF  $Y \notin X_G^+$  THEN 输出 " $F^+ \neq G^+$ ", 停止.

2. 输出 " $F^+ = G^+$ ", 停止.

### 算法思想:

由于  $G \subseteq F$ ,  $G \subseteq F^+$ , 不需要判别. 所以, 要判定是否 " $F^+ = G^+$ ", 只需要判别是否 " $F \subseteq G^+$ ".





## • 函数依赖保持性的判别

[例] 给定关系模式  $R\langle U, F \rangle$ , 其中  $U=\{A, B, C, D\}$ ,  
 $F=\{A \rightarrow B, B \rightarrow C, C \rightarrow D, D \rightarrow A\}$ , 判断关系模式  $R$  的分解  
 $\rho=\{AB, BC, CD\}$  是否具有依赖保持性。

解: 因为  $\Pi_{AB}(F)=\{A \rightarrow B, B \rightarrow A\}$

$\Pi_{BC}(F)=\{B \rightarrow C, C \rightarrow B\}$

$\Pi_{CD}(F)=\{C \rightarrow D, D \rightarrow C\}$

$\Pi_{AB}(F) \cup \Pi_{BC}(F) \cup \Pi_{CD}(F)$

$=\{A \rightarrow B, B \rightarrow A, B \rightarrow C, C \rightarrow B, C \rightarrow D, D \rightarrow C\}$

从中可以看到  $A \rightarrow B, B \rightarrow C, C \rightarrow D$  均得以保持,

又因为  $D^+ = ABCD, A \subseteq D^+$ , 所以  $D \rightarrow A$  也得以保持,

所以该分解具有依赖保持性。





## • 关系模式分解算法

- 分解具有无损连接性和分解保持函数依赖是两个互相独立的标准。具有无损连接性的分解不一定能够保持函数依赖。同样，保持函数依赖的分解也不一定具有无损连接性。





# 关系模式分解算法

- 3NF 分解算法1(达到3NF且保持函数依赖的分解)
  - 输入:关系模式 $R(U,F)$ , $G$ 是极小函数依赖集,  $R$ 是1NF。
  - 输出:具有函数依赖保持性的分解 $\rho$ ,  $\rho$ 中所有关系模式都是3NF。
  - 算法:
    - (1)  $\rho := \text{空集合};$
    - (2) IF 存在 $X \rightarrow A \in G$ 且 $\{X,A\}=U$  THEN  $\rho := \{R\}$ ,停止。
    - (3) ELSE FOR每个不出现在 $G$ 的任何一个函数依赖中的属性 $A$   
DO  $\rho := \rho \cup \{R(A)\};$
    - (4) FOR  $X \rightarrow A \in G$  DO  $\rho := \rho \cup \{R(X,A)\};$
    - (5) ENDFOR;
    - (6)ENDIF。





• 如果函数依赖集F满足下列条件，则称F为一个极小函数依赖集。亦称为最小依赖集或最小覆盖：

- F中任一函数依赖的右部仅含有一个属性
- F中不存在这样的函数依赖 $X \rightarrow A$ ，使得F与 $F - \{X \rightarrow A\}$ 等价
- F中不存在这样的函数依赖 $X \rightarrow A$ ，X有真子集Z使得 $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ 与F等价

[例] 设有关系模式 $R \langle U, F \rangle$ ，其中 $U = \{C, T, H, R, S, G\}$ ，其中C表示课程，T表示教师，H表示时间，R表示教室，S表示学生，G表示成绩。

函数依赖集F及其所反映的语义分别为：

$F = \{CS \rightarrow G, C \rightarrow T, TH \rightarrow R, HR \rightarrow C, HS \rightarrow R\}$ 。

问：将其保持函数依赖分解为3NF。

解：(1) 求出F的最小依赖集：

$F_m = \{CS \rightarrow G, C \rightarrow T, TH \rightarrow R, HR \rightarrow C, HS \rightarrow R\}$

(2)  $\rho = \{CSG, CT, THR, HRC, HSR\}$





# 关系模式分解算法

## • 3NF分解算法2 (达到3NF且具有无损连接和保持函数依赖的分解)

- 输入：关系模式 $R(U, F)$ ,  $F$ 是极小函数依赖集。
- 输出：具有无损连接和函数依赖保持性的分解 $\tau$ ,  $\tau$ 中所有关系模式都是3NF。
- 算法：

(1)调用算法1产生 $R$ 的分解 $\rho = \{R_1, R_2, \dots, R_n\}$

(2)构造分解 $\tau = \{R_1, R_2, \dots, R_n, \mathbf{R_k}\}$ , 其中 $R_k$ 是由 $R$ 的一个候选键 $K$ 构成的关系。





$F = \{CS \rightarrow G, C \rightarrow T, TH \rightarrow R, HR \rightarrow C, HS \rightarrow R\}.$

[例]对于上例的关系模式，将其无损连接和依赖保持分解为3NF。

由上例求出依赖保持性分解为：

$\rho = \{CSG, CT, THR, HRC, HSR\}$

$(HS)^+ = U$ , HS为候选键,

由于HSR包含HS,所以

由上例求出无损连接性分解为：

$\rho = \{CSG, CT, THR, HRC, HSR\}$





# 关系模式分解算法

## • BCNF分解算法

- 输入：关系模式 $R(U, F)$
- 输出：具有无损连接的分解 $\tau$ ,  
 $\tau$ 中所有关系模式都是BCNF。

— 算法：

(1)  $\tau = \{R\}$

(2) WHILE  $\tau$  中存在非BCNF关系模式 DO

(3)     选择一个非BCNF模式  $S \in \tau$ ;

(4)     选择 $S$ 的违反BCNF要求的依赖  $X \rightarrow Y$ ;

(5)     用关系模式 $(S-Y)$ 和 $(X \cup Y)$ 取代 $\tau$ 的 $S$

(6) ENDWHILE



# 练习



例1, 设有关系模式 $R(U, F)$ , 其中 $U=ABC$ ,

$F=\{A \rightarrow \{B, C\}, B \rightarrow C, A \rightarrow B, \{A, B\} \rightarrow C\}$ , 求 $F_{\min}$

(1) 将 $F$ 中所有的函数依赖的依赖关系因素写成单属性集形式:  $G=\{A \rightarrow B, A \rightarrow C, B \rightarrow C, A \rightarrow B, \{A, B\} \rightarrow C\}$

这里多出一个 $A \rightarrow B$ , 可以删除, 得到

$G=\{A \rightarrow B, A \rightarrow C, B \rightarrow C, \{A, B\} \rightarrow C\}$

(2)  $G$ 中的 $A \rightarrow C$ 可以从 $A \rightarrow B$ 和 $B \rightarrow C$ 推导出,  $A \rightarrow C$ 冗余, 删除。

$G=\{A \rightarrow B, B \rightarrow C, \{A, B\} \rightarrow C\}$

(3)  $G$ 中的 $\{A, B\} \rightarrow C$ 可以从 $B \rightarrow C$ 推导出来, 是冗余的, 删除。

$G=\{A \rightarrow B, B \rightarrow C\}$

所以,  $F_{\min} = \{A \rightarrow B, B \rightarrow C\}$



## 例2、判定

$S(sno, sname, sdept, sage), C(cno, cname, cpno),$   
 $SC(sno, cno, grade)$ 的范式等级。

解：S关系的候选键为sno,

函数依赖为 $sno \rightarrow sname, sno \rightarrow sdept, sno \rightarrow sage,$

非键属性sname,sdept,sage对键sno没有部分或传递函数依赖。

所以 $S \in 3NF$ 。

而且决定因素都是键，所以 $S \in BCNF$ 。

同理，C关系和SC关系都属于BCNF



# 练习

例3、设有关系模式project(工程号,材料号,数量,开工日期,完工日期,单价).

语义:每项工程可以使用多种材料,每种材料可以供多项工程使用;每项工程开工前必须确定开工日期和完工日期;每种材料有确定的价格.

请写出函数依赖,并判断该关系模式的范式等级,分析是否存在操作异常?并分解为高一级范式。





project(工程号,材料号,数量,开工日期,完工日期,单价).

解:

候选键为 (工程号, 材料号),

函数依赖为:

工程号  $\rightarrow$  开工日期;

工程号  $\rightarrow$  完工日期; 材料号  $\rightarrow$  单价

(工程号, 材料号)  $\rightarrow$  数量;

(工程号, 材料号)  $\xrightarrow{p}$  开工日期;

(工程号, 材料号)  $\xrightarrow{p}$  完工日期;

(工程号, 材料号)  $\xrightarrow{p}$  单价







**project(工程号,材料号,数量,开工日期,完工日期,单价).**

候选键为（工程号，材料号），

函数依赖为：工程号 → 开工日期；工程号 → 完工日期；材料号 → 单价  
（工程号，材料号） → 数量；（工程号，材料号） → 开工日期；  
（工程号，材料号） → 完工日期；（工程号，材料号） → 单价

**因为存在非键属性对键的部分函数依赖,所以,**

- (1) 如果工程项目确定后，若暂时未用到材料，则该工程的数据因缺少关键字的一部分材料号，而不能进入到数据库中，出现插入异常。**
- (2) 工程下马，则删去该工程的操作也可能丢失材料方面的信息。**

**分解后project1(工程号,材料号,数量)**

**project2(工程号,开工日期,完工日期)**

**project3(材料号,单价)**





# 练习

**例4、设关系R (CN,TN,TD) 其中CN表示课程名，TN表示教师名字（没有重名教师），TD表示教师地址。语义：一门课只被一个教师讲，一个教师可以讲多门课。判断为几范式？为什么？是否存在删除异常？分解为高一级范式。**





## $R(CN, TN, TD)$

解：R的候选键为CN；

$CN \rightarrow TN, TN \not\rightarrow CN, TN \rightarrow TD$ , 所以  $CN \rightarrow TD$

因为存在非键属性对键的传递函数依赖，  
所以  $R \notin 3NF$ 。

又因为不存在非键属性对键的部分函数依赖，  
所以  $R \in 2NF$ 。

存在删除异常，当删除某课时会删除教师的有关信息。

分解后  $R1(CN, TN); R2(TN, TD)$





# 练习

例5、关系模式R(职工号,职工名,年龄,性别,部门号,部门名),职工名允许重复

R是否属于3NF? 为什么? 对之进行规范化。

解: R的候选键为职工号;

函数依赖为:

职工号 $\rightarrow$ 部门号, 部门号 $\rightarrow$ 职工号, 部门号 $\rightarrow$ 部门名;

存在非键属性对键的传递函数依赖, 所以 $R \notin 3NF$ 。

不存在非键属性对键的部分函数依赖, 所以 $R \in 2NF$ 。

分解后

R1(职工号,职工名,年龄,性别)

R2(部门号,部门名)



# 逻辑数据库设计

## • 逻辑数据库设计的步骤

- 形成初始关系数据库模式;
- 关系模式规范化;
- **关系模式优化;**
- 定义关系上的完整性和安全性约束;
- 子模式定义;
- 性能估计。





## 4.3.3 关系模式优化

- 关系模式的优化是根据需求分析和概念设计中定义的**事务的特点**，对关系进行分解，提高数据操作的效率和存储空间利用率。
- 关系数据模型的优化通常以规范化理论为指导。





## 4.3.3 关系模式优化

- 常用的关系分解方法

- 水平分解

- 什么是水平分解

- 把(基本)关系的元组分为若干子集合, 定义每个子集合为一个子关系, 以提高系统的效率。

- 使用范围

- 满足“80/20原则”的应用
        - » 在一个大型关系中, 经常被使用的数据知识是很有限的一部分
        - » 可以把经常使用的数据分离出来, 形成一个子关系
      - 并发事务经常存取不相交的数据
        - » 如果关系R上有n个事务, 而且多数事务存取的数据不相交, 则R可以分解为少于或等于n个子关系, 是每个事务存取的数据形成一个关系





- 例如，一个大学的数据库系统中包含：  
学生关系模式(Sno, Sdept, Sname, Sage,...)

S1	计算机	张三		
S2	计算机	李四		
S3	计算机	王五		
S4	计算机			
S5	计算机			
...	....			
...				
...	物理			
...	物理			
...	物理			
...	物理			
...	...			
...	机电			
...	机电			
...	机电			
...	机电			
...	...			
...	材料			
...	材料			
...	.....			



S1	计算机	张三		
S2	计算机	李四		
S3	计算机	王五		
S4	计算机			
S5	计算机			
...	....			
...				
...	计算机			
...	计算机			
...	计算机			
...	计算机			
...	...			

...	...			
...	机电			
...	机电			
...	机电			
...	机电			
...	...			
...	机电			
...	机电			
...	.....			

...	物理			
...	物理			
...	物理			
...	物理			
...	...			
...	物理			
...	物理			
...	物理			
...	物理			
...	...			







## 4.3.3 关系模式优化

- 常用的关系分解方法

- 垂直分解

- 垂直分解使用范围

- 取决于分解后R上的所有事务的总效率是否得到了提高

- 垂直分解方法

- 简单情况：直观分解

- 复杂情况：用关系模式分解算法

垂直分解必须不损失关系模式的语义  
(保持无损连接性和保持函数依赖)



教师关系模式(职工号,姓名,性别,住址,出生日期,婚姻状况,学历,学位,政治面貌,职称,职务,工资,工龄,教学效果)

人事部门:

教师关系模式1(职工号,姓名,性别,住址,出生日期,婚姻状况,政治面貌)

教务部门:

教师关系模式2(职工号,姓名,学历,学位,职称,教学效果)

财务部门:

教师关系模式3(职工号,姓名,工资,工龄,职务)





# 逻辑数据库设计

## • 逻辑数据库设计的步骤

- 形成初始关系数据库模式;
- 关系模式规范化;
- 关系模式优化;
- 定义关系上的完整性和安全性约束;
- 子模式定义;
- 性能估计。



## 4.3.4 定义关系上的完整性和安全性约束

- 每个关系模式上的完整性约束分为三类：
  - 属性上的完整性约束
  - 多个属性间的完整性约束
  - 不同关系模式的属性间的完整性约束
- 安全性约束分两类：
  - 属性上的安全性约束
  - 关系模式上的安全性约束





# 逻辑数据库设计

## • 逻辑数据库设计的步骤

- 形成初始关系数据库模式;
- 关系模式规范化;
- 关系模式优化;
- 定义关系上的完整性和安全性约束;
- 子模式定义;
- 性能估计。





## 4.3.5 子模式定义

- 利用视图定义外模式
  - 使用更符合用户习惯的列名（属性重命名）
  - 为不同级别的用户定义不同的视图
  - 简化用户对系统的使用





例：

教师关系模式中包括职工号、姓名、性别、出生日期、婚姻状况、学历、学位、政治面貌、职称、职务、工资、工龄、教学效果等属性。

学籍管理应用只能查询教师的职工号、姓名、性别、职称数据；

课程管理应用只能查询教师的职工号、姓名、性别、学历、学位、职称、教学效果数据；

教师管理应用则可以查询教师的全部数据。



教师关系模式(职工号,姓名,性别,出生日期,婚姻状况,学历,学位,政治面貌,职称,职务,工资,工龄,教学效果)

授权教师管理应用能访问教师表

定义两个外模式:

授权课程  
管理应用

教师\_学籍管理(职工号, 姓名, 性别, 职称)

授权学籍  
管理应用

教师\_课程管理(工号, 姓名, 性别, 学历,  
学位, 职称, 教学效果)

这样就可以防止用户非法访问本来不允许他们查询的数据, 保证了系统的安全性。





# 逻辑数据库设计

## • 逻辑数据库设计的步骤

- 形成初始关系数据库模式;
- 关系模式规范化;
- 关系模式优化;
- 定义关系上的完整性和安全性约束;
- 子模式定义;
- 性能估计。





## 4.3.6 性能估计

- 性能估计是对已经设计完成的逻辑数据库的**时间复杂性**和**空间复杂性**进行估算，其结果可以用来检验现有的计算机软硬件环境是否满足要求，以便调整软硬件环境或数据库的设计。
- 三个测度
  - 逻辑记录存取数
  - 信息传输量
  - 存储空间占用量





设 $T_1, T_2, \dots, T_n$ 是逻辑数据库上运行的 $n$ 个事务  
 $f_1, f_2, \dots, f_n$ 是这 $n$ 个事务的发生频率

- 逻辑记录存取数的估算：

$$LRA = \sum_{j=1}^m \sum_{i=1}^n f_i \times LRA_{ij}$$

其中， $LRA_{ij}$ 是第 $i$ 个事务访问  
第 $j$ 个关系的逻辑记录数

- 信息传输量的估算：

$$LIA = \sum_{j=1}^m \sum_{i=1}^n LRA_{ij} \times f_i \times RECSIZE_j$$

其中， $RECSIZE_{ij}$ 是第 $j$ 关系  
的记录长度

- 存储空间占用量的估算：

$$DSTOR = \sum_{j=1}^m RECSIZE_j \times NREC_j$$

其中， $NREC_j$ 是第 $j$ 关  
系的记录个数





## 4.4 物理数据库设计





# 物理数据库设计

- 设计任务

- 在逻辑数据库设计基础上，为每个关系模式选择合适的**存储结构**和**存取方法**，使得数据库上的事务能够高效率的运行

- 设计步骤

- 分析影响物理数据库设计的因素
- 为关系模式选择存取方法
- 设计关系、索引等数据库文件的物理存储结构





## • 物理数据库设计步骤

- 分析影响物理数据库设计的因素
- 为关系模式选择存取方法
- 设计关系、索引等数据库文件的物理存储结构





# 影响物理数据库设计的因素

- 对于数据库**查询事务**，需得到如下信息：

- (1) 查询的关系；
- (2) 查询条件所涉及的属性；
- (3) 连接条件所涉及的属性；
- (4) 查询的投影属性。

例如，关系R更新频率很高，则R上的索引等要尽可能少

- 对于数据**更新事务**，需得到如下信息：

- (1) 被更新的关系；
- (2) 每个关系上的更新操作的类型；
- (3) 删除和修改操作条件所涉及的属性；
- (4) 修改操作要改变的属性值。

- 了解每个事务在各关系上运行的**频率**

- 了解每个事务的**时间约束**

上述信息是我们确定关系的存取方法的依据



## • 物理数据库设计步骤

- 分析影响物理数据库设计的因素
- 为关系模式选择存取方法
- 设计关系、索引等数据库文件的物理存储结构







# 为关系模式选择存取方法

- 常用的存取方法可以分为三类：
  - 索引方法
  - HASH方法
  - 聚簇方法





# 为关系模式选择存取方法

## • 索引存取方法的选择

— 根据在R上事务 $T_1$ 、 $T_2$ 、...、 $T_k$ 的信息确定候选索引，规则如下：

- 如果一个(或一组)属性经常在操作条件中出现，则考虑在这个(或这组)属性上建立索引；
- 如果一个属性经常作为最大值和最小值等聚集函数的参数，则考虑在这个属性上建立索引；
- 如果一个(或一组)属性经常在连接操作的连接条件中出现，则考虑在这个(或这组)属性上建立索引；
- 如果一个(或一组)属性经常作为投影属性使用，则考虑在这个(或这组)属性上建立索引；





# 为关系模式选择存取方法

- 索引存取方法的选择

- 一个关系上可以建立多个索引
- 优化配置索引

- 不加索引?
- 一个索引
- 两个索引?
- ...

最小化 $\text{Cost}(R)$

$$\text{cost}(R) = \sum_{i=1}^n f_i \text{cost}(T_i)$$

其中, (1)  $\text{Cost}(T_i)$  是事务  $T_i$  的代价  
(2)  $f_i$  是  $T_i$  发生的频率





## • 物理数据库设计步骤

- 分析影响物理数据库设计的因素
- 为关系模式选择存取方法
- 设计关系、索引等数据库文件的物理存储结构





# 物理存储结构设计

- 确定如何在磁盘存储器上存储关系、索引和聚簇，使得空间利用率最大化，数据操作引起的系统开销最小化
- 与具体数据库管理系统相关

— 例如 Oracle





# Oracle物理数据库设计

- **存储记录**

- 最基本数据单位。

- **物理存储块**

- **数据域**

- 由一个或多个连续的物理存储块组成的存储空间。

- **数据段**

- 与关系、索引、聚集等数据库对象相对应
  - 由一个或多个数据域构成。
  - 构成数据段的数据域可以具有相同或不同的长度。

- **INITIAL-EXETENT:** 数据段初始数据域的容量。当数据段建立时, 系统立即分配 INITIAL-EXETENT量的空间。

- **NEXT-EXETENT:** 第二个数据域的容量。

- **MINEXTENTS:** 数据段的最小数据域数。当数据段建立时系统自动为该数据段分配MINEXTENTS数据域。

- **PCTINCREASE:** 由第三个数据域开始, 每个数据域增长的百分比。对于 $K \geq 2$ , 设第 $k$ 个数据域容量为 $V$ , 第 $k+1$ 数据域的容量为 $(1+PCTINCREASE) \times V$ 。





- ORACLE数据库物理存储块的设计:

- 参数PCTFREE的确定:

- PCTFREE是数据块内为存储记录的扩展而保留的空间百分比。如果某数据块的自由空间的百分比低于PCTFREE, 该数据块将移出自由链, 无新记录可加入此块, 但此块中的记录可以扩展。

- 参数PCTUSED的确定

- PCTUSED是数据块中数据量不得低于的百分比。如果某数据块所存储的数据的百分比低于PCTUSED, 该数据块将加入自由链, 使新记录可加入此块。

- $PCTFREE + PCTUSED \leq 100\%$ 。





# 小结

- 概念数据库设计:

- 实体联系模型、ER模型
- 实体、实体集
- 实体的属性、实体的属性值、复合属性、单值属性、多值属性、导出属性、空值
- 候选键、简单键、复合键
- 实体间的联系
- 弱实体、识别实体、识别联系
- ER图







- 逻辑数据库设计:

- 形成初始关系模式
- 函数依赖、完全函数依赖、部分函数依赖、传递函数依赖
- Armstrong公理系统、三条推理规则
- 求属性闭包、求候选键
- 两个函数依赖集等价的判定、求最小函数依赖集
- 关系模式的规范形式
  - 1NF、2NF、3NF、BCNF
- 关系模式的规范化方法
  - 无损连接性、函数依赖保持性、判别方法
  - 关系模式的分解算法



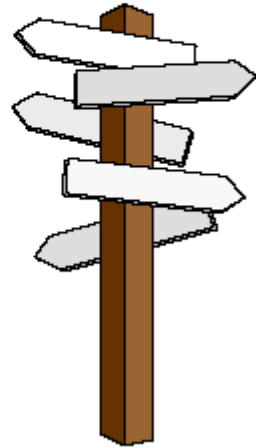


- 物理数据库设计的步骤





**Now let's go to  
Next Chapter**





# • 基本概念(续)

## — 约束

### • 实体对应约束

— 1:1, 1:n, n:1, n:n

### • 实体关联约束

— 全域关联约束

— 部分关联约束

» 由于并非每个教师都是系主任，所以教师实体型上有一个与联系T\_D\_2有关的实体关联约束：教师实体型中仅有部分实体与系实体相关联。这种实体关联约束称为**部分关联约束**。





## • 函数依赖

### — 定义4:

- 设 $R$ 是一个具有属性集合 $U$ 的关系模式,  $K \subseteq U$ 。若 $K \rightarrow U$ , 则称 $K$ 是 $R$ 的一个**超码**(superkey)
- 超码可以唯一地识别关系的元组。
- 设 $R$ 是一个具有属性集合 $U$ 的关系模式,  $K \subseteq U$ 。如果 $K$ 是 $R$ 的一个超码, 且不存在 $K$ 的真子集 $Z$ 使得 $Z \rightarrow U$ 。则称 $K$ 是 $R$ 的一个**候选键**(候选码/码/键):
- 一个关系模式中可能具有多个候选键, 指定其中的一个作为识别关系元组的**主键**(主码)。
- 包含在任何一个候选键中的属性称为**键属性**。
- 不包含在任何候选键中的属性称为**非键属性**。
- 在最简单的情况下, 候选键只包含一个属性。
- 在最复杂的情况下, 候选键包含关系模式的所有属性, 称为**全键**。





- 函数依赖

- 定义5:

- 设 $X$ 是关系模式 $R$ 的属性子集合。如果 $X$ 是另一个关系模式的候选键，则称 $X$ 是 $R$ 的外键。





- 第一范式

- 设 $R$ 是一个关系模式。如果 $R$ 的每个属性的值域都是不可分的简单数据项的集合，则称 $R$ 为**1NF**。

- 第二范式

- 若 $R$ 是1NF，且每一个非键属性都完全函数依赖于 $R$ 的候选键(码)，则 $R$ 称为**2NF**。

- 第三范式

- 如果 $R$ 是2NF，且它的任何一个非键属性都不传递地依赖于任何候选键，则 $R$ 称为**3NF**。

- BC范式

- 设 $R$ 是1NF。如果对于 $R$ 的每个函数依赖 $X \rightarrow Y$ ，则 $X$ 必为候选键，则 $R$ 是**BCNF**。

