第3章 数据库设计

清华大学计算机系 李国良

liguoliang@tsinghua.edu.cn
http://dbgroup.cs.tsinghua.edu.cn/ligl

本章知识框架

现实世界
(如学生信息管理)

学生有学号、姓名、班级等信息，每个学生可以选修多门课程...

需求分析

数据库管理员

数据库设计核心步骤

- 现实世界需求分析
- 概念结构设计阶段
- 逻辑结构设计阶段
- 物理结构设计阶段
- 数据库建设和维护

数据模型

概念数据模型	实体-联系模型
逻辑数据模型	关系模型
物理数据模型	存储结构设计

规范化设计

规范化设计理论和实现

规范化程度

- 第一范式
- 第二范式

提高

规范化实现

- 关系模式的分解

理论基础

判定

- 函数依赖
- 数据依赖的公理系统

规范化设计理论

目录

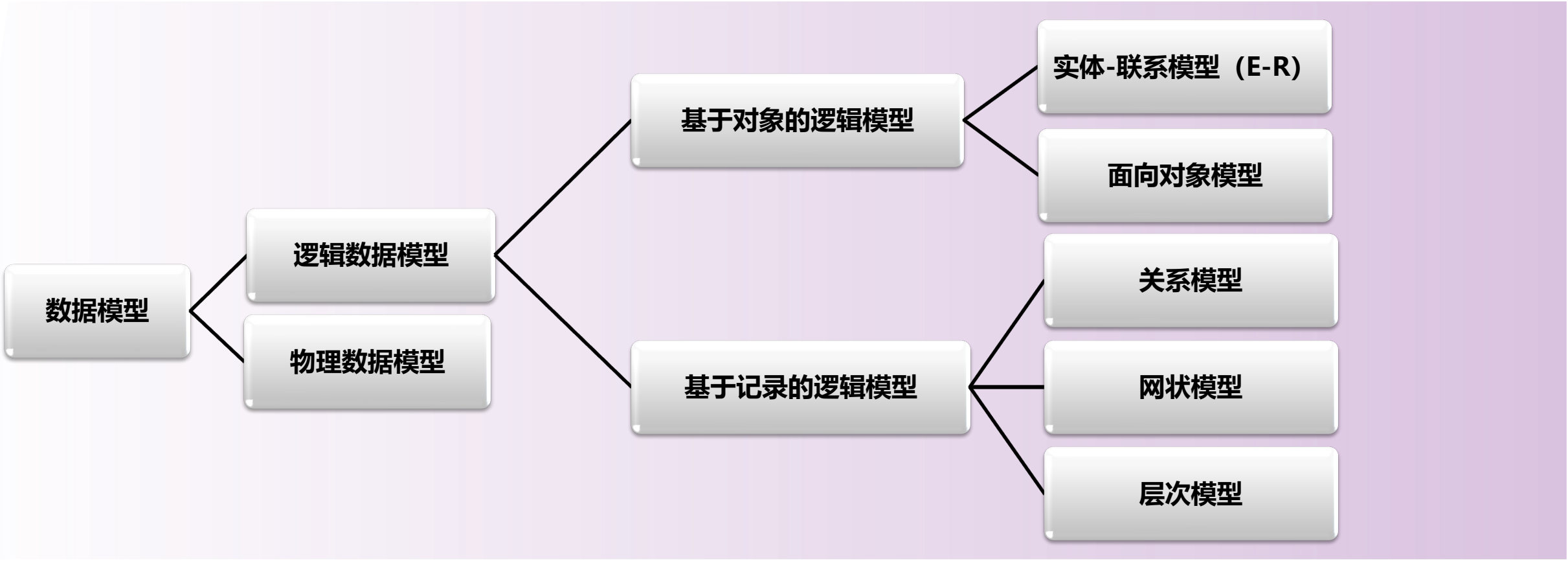
数据库设计和数据模型

- 数据库是组织、存储和管理数据的集合
- 数据库设计会直接影响数据库自身和上层应用的性能
- 一个好的数据库设计可以提高存储空间的利用率和数据存取的效率，可以更好地支持基于数据库的应用系统
- 在数据库领域，数据模型被广泛用于表示这些数据库设计的“描述”，更好的刻画数据

数据库设计和数据模型（续）

➢ 数据模型概述

- 数据库结构的基础是数据模型，它是描述数据(数据结构)、数据之间的联系，数据语义即数据操作，以及一致性(完整性)约束的**概念和工具的集合**。



数据库设计和数据模型（续）

概念设计阶段

□ 概念数据模型

- 将现实世界的客观事物及其关系抽象为“实体”和“关系”等形式，用于描述业务领域的数据对象及其关系。
- 概念数据模型主要是依据用户对现实业务的理解来对数据对象进行建模，主要用于数据库的概念设计。

逻辑设计阶段

□ 逻辑数据模型

- 在概念数据模型的基础上，需要进一步考虑这些数据对象在计算机系统中的逻辑表示，主要用于数据库管理系统的逻辑设计。
- 常见的逻辑数据模型有关系模型、层次模型和网状模型等。

物理设计阶段

□ 物理数据模型

- 该模型已经将数据抽象为“表格”、“列”、“键”等形式，该数据模型需要具体考虑数据对象如何在数据库管理系统中物理实现。
- 数据表、索引的设计。

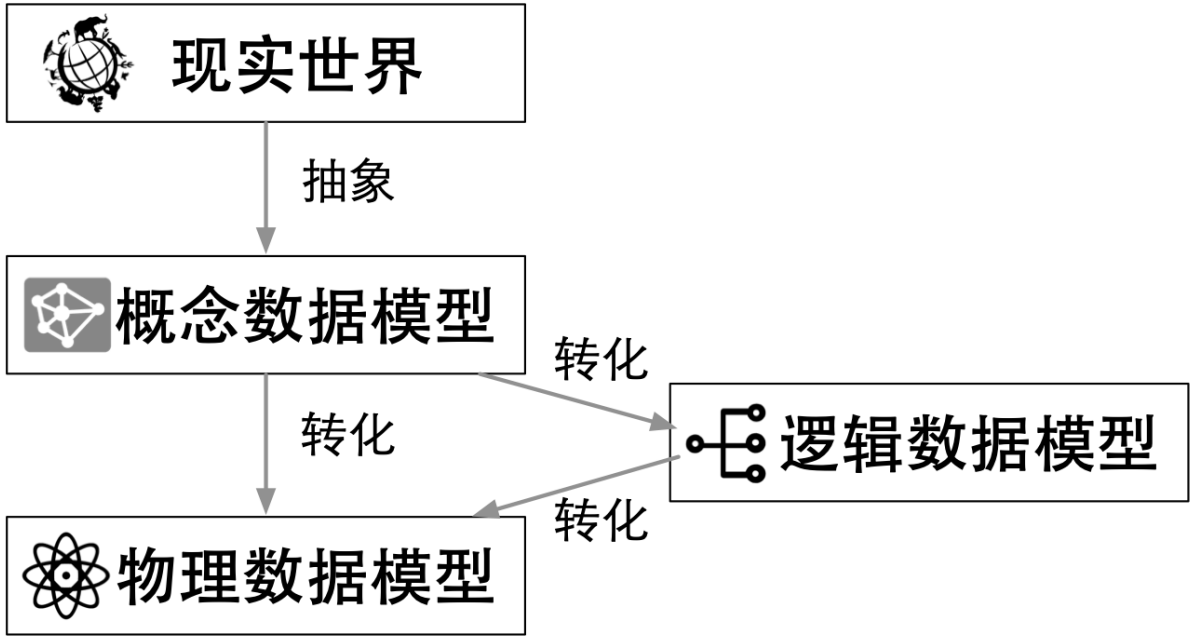
数据库设计模型转换（续）

- **数据模型元素对应关系**

概念数据模型	逻辑数据模型	物理数据模型
实体（entity）	实体（entity）	表（table）
---	属性（attribute）	列（column）
---	标识符(primary/foreign identifier)	键（primary/foreign key）
关系（relation）	关系（relation）	参照完整性约束(reference)

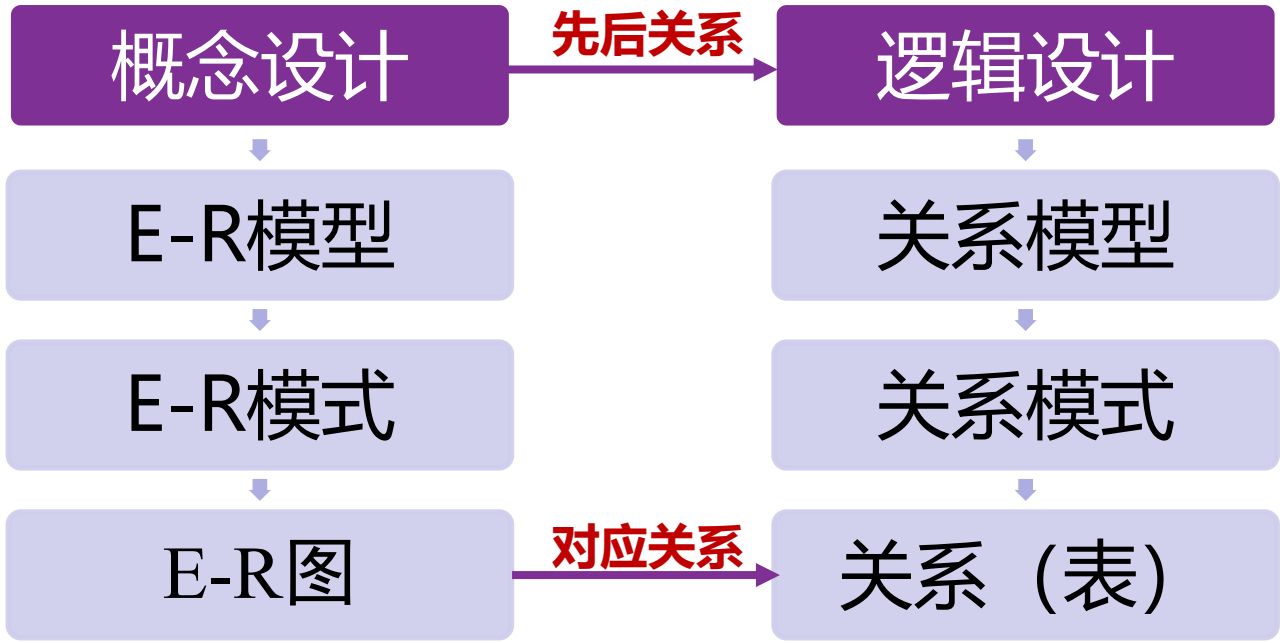
数据库设计模型转换（续）

- 由于E-R模型都可以进行概念设计和逻辑设计阶段的数据模型表示，
  - 无论采用何种设计路线，最终的目标都是将E-R模型转化为物理数据模型。
- 在关系数据库的设计中，物理数据模型即是关系模型。
  - 因此，**关系数据库设计的一个核心步骤就是将E-R模型转化为关系模型**。



数据库设计模型转换（续）

E-R图转化为关系



目录

概念结构设计：E-R模型

- **E-R模型是实体-联系模型（entity-relationship model）的简称**
  - 是用于描述现实世界的概念数据模型
  - 也可以用于表示关系数据库的结构
  - 由美籍华人计算机科学家陈品山（Peter Pin-Shan Chen）提出
  - 在数据库设计领域有着广泛的应用目前大部分的数据库设计工具和产品均采用E-R模型进行数据库的概念结构设计

E-R模型

2 E-R模型

2.1 E-R模型的基本元素

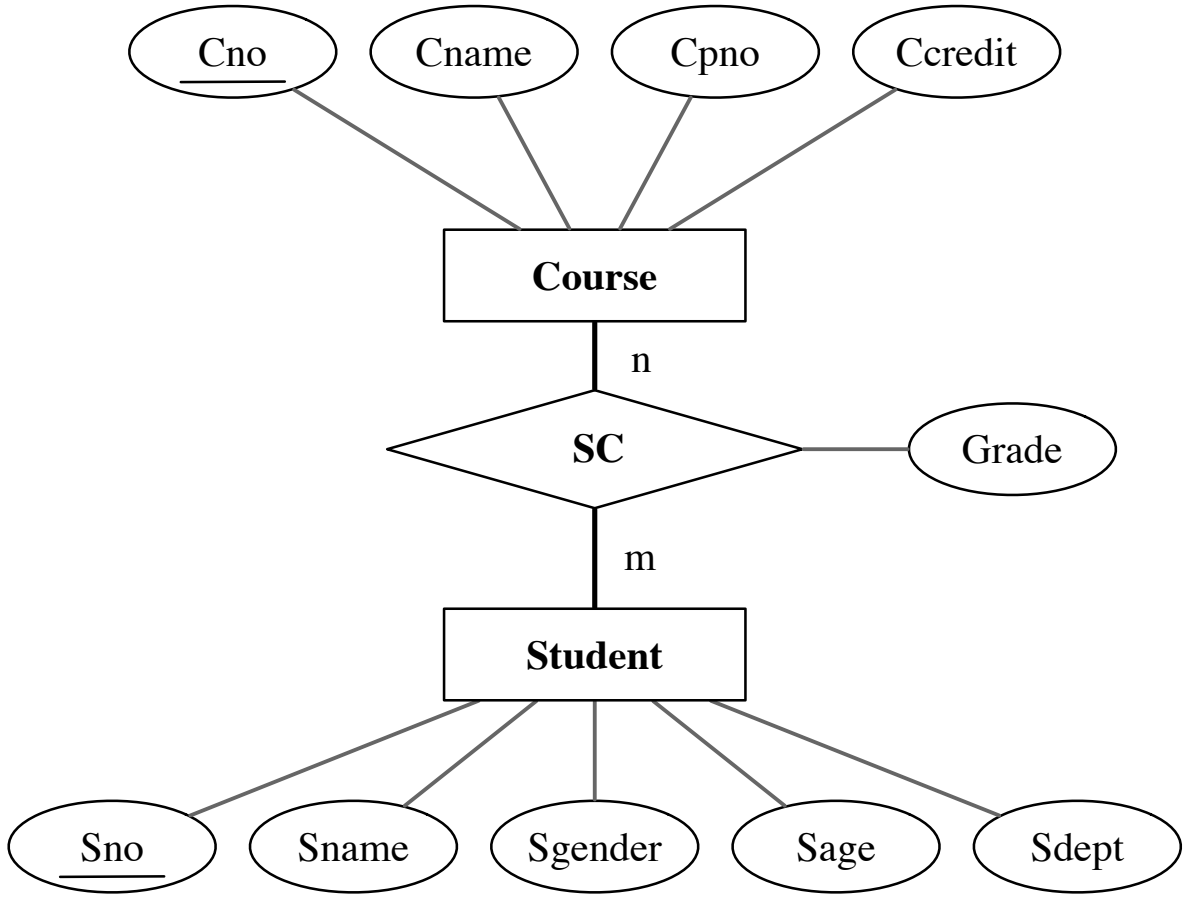
- 实体和实体集
- 属性
- 联系

2.2 E-R图

2.3 E-R联系类型

E-R模型的基本元素

- 实体和实体集
- 属性
- 联系



E-R模型的基本元素（续）

➢ 实体和实体集

- **实体**是对现实世界中事物数据概念的某种抽象。
  - ◆通俗地说，实体可以指现实世界的人、物或抽象的概念等。
  - ◆例如，一个人是实体，一个公司也是实体。

- 多个具有相同性质的同类实体构成的集合，称为**实体集**。
  - ◆例如，一所大学的所有学生是一个实体集，特定的实体称为实体实例。

E-R模型的基本元素（续）

➢ 属性

- 实体集都可以被一组特征来描述，这些用来描述实体集的数据特征被称为实体集的**属性**。
- 通常不同实体集的属性是不同的。
- 例如，下表的Student实体集有Sno、Sname、Sgender、Sage和Sdept属性。

Student表

Sno（学号）	Sname（姓名）	Sgender（性别）	Sage（年龄）	Sdept（所在系）
2021310721	李博	男	17	CS
2021310722	赵宇	男	19	CS
2021310723	张敏	女	18	CS
2021310724	王勇	男	18	MA
2021310725	刘佳	女	17	MA

E-R模型的基本元素（续）

- **标识符**：可以唯一标识不同的实体集的某一属性
  - 例如，Student中的Sno属性可以唯一标识不同学生，它可以作为Student实体集的标识符。
- **复合标识符**：当必须选取多个属性的组合来作为实体的唯一标识，称该属性的组合为复合标识符。
  - 例如，SC实体集中，如果仅用Cno作为标识符，则不能区分不同学生（Sno）的选课信息。同理，也找不到其它单个属性用于唯一标识SC实体集。
  - 只有将Sno和Cno组合在一起，才能唯一标识SC实体集，因此Sno和Cno是其复合标识符。

Student表

Sno（学号）	Sname（姓名）	Sgender（性别）	Sage（年龄）	Sdept（所在系）
2021310721	李博	男	17	CS
2021310722	赵宇	男	19	CS

SC表

Sno（学号）	Cno（课程号）	Grade（成绩）
2021310721	5	98
2021310722	1	87
2021310723	1	92

E-R模型的基本元素（续）

- **联系**
  - 正如现实世界中事物之间都有某种联系一样，这些事物在数据库中也必然存在联系
  - 这些联系可以是实体与实体之间的联系，如学生实体和课程实体之间的“选课”联系
  - 也可以是实体内部的联系，如学生实体与其各属性之间的联系
- E-R模型中的联系主要指代实体与实体之间的联系
  - 这种联系一般是用动词来命名。如，学生实体与课程实体之间的一种联系是“选课”

Student表

Sno（学号）	Sname（姓名）	Sgender（性别）	Sage（年龄）	Sdept（所在系）
2021310721	李博	男	17	CS
2021310722	赵宇	男	19	CS
2021310723	张敏	女	18	CS
2021310724	王勇	男	18	MA
2021310725	刘佳	女	17	MA

Course表

Cno（课程号）	Cname（课程名）	Cpno（先修课）	Ccredit（学分）
1	数据库	2	4
2	数据结构与算法	6	4
3	操作系统	2	3
4	高等数学		4
5	软件工程	6	2
6	程序设计		3

E-R模型

2 E-R模型

2.1 E-R模型的基本元素

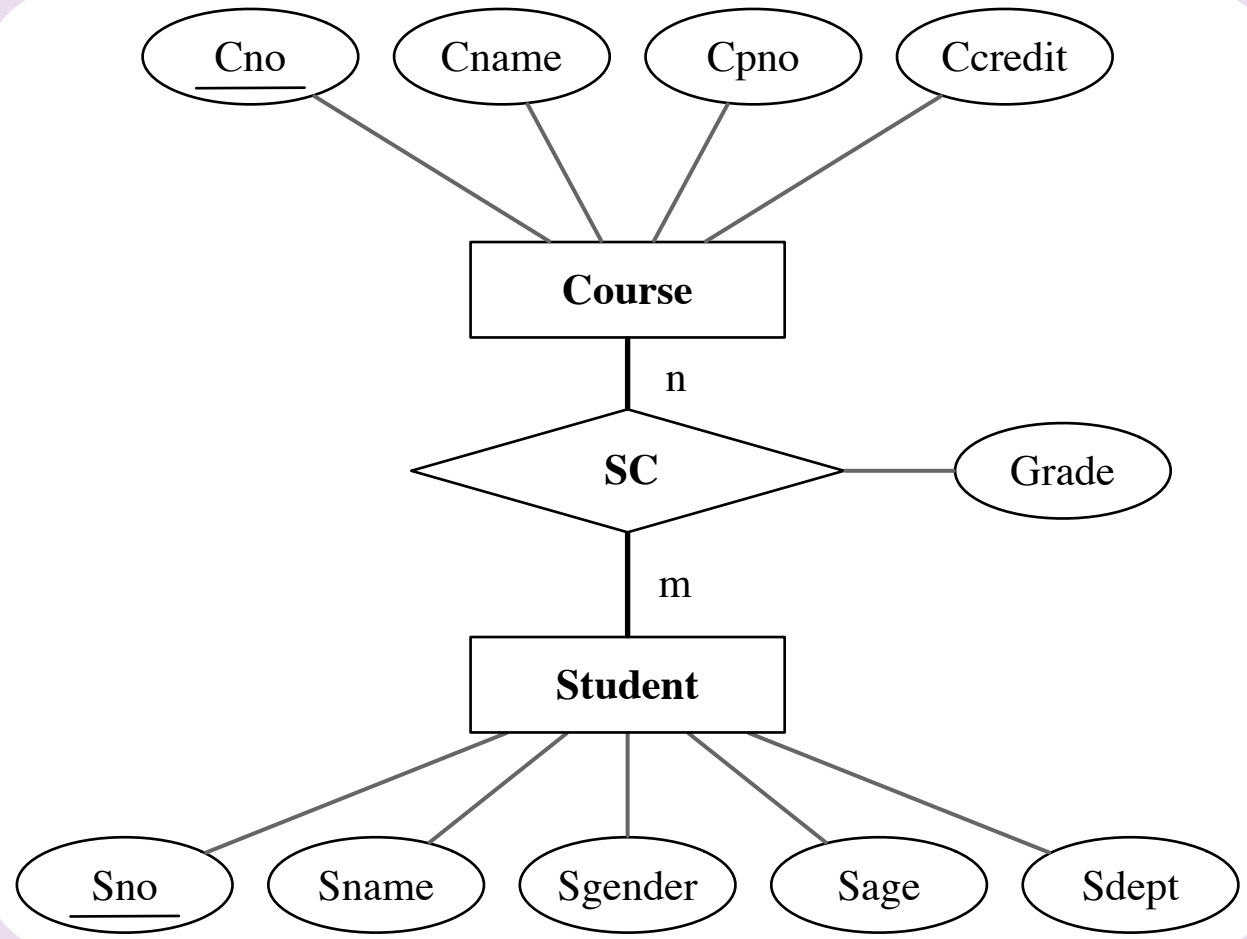
- 实体和实体集
- 属性
- 联系

2.2 E-R图

2.3 E-R联系类型

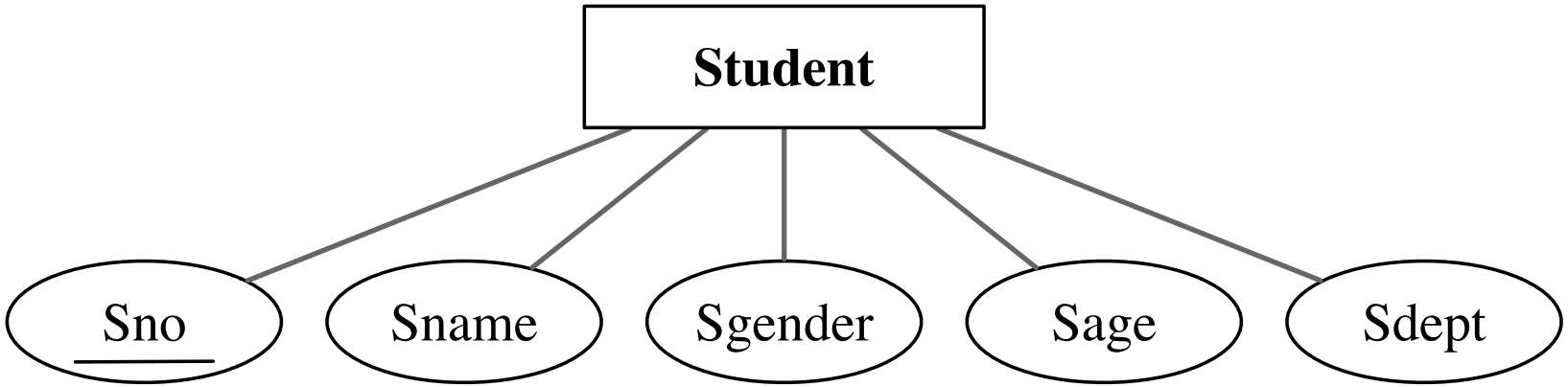
E-R图三要素

- **E-R图**
  - 实体集的表示方法
  - 属性的表示方法
  - 联系的表示方法



E-R图

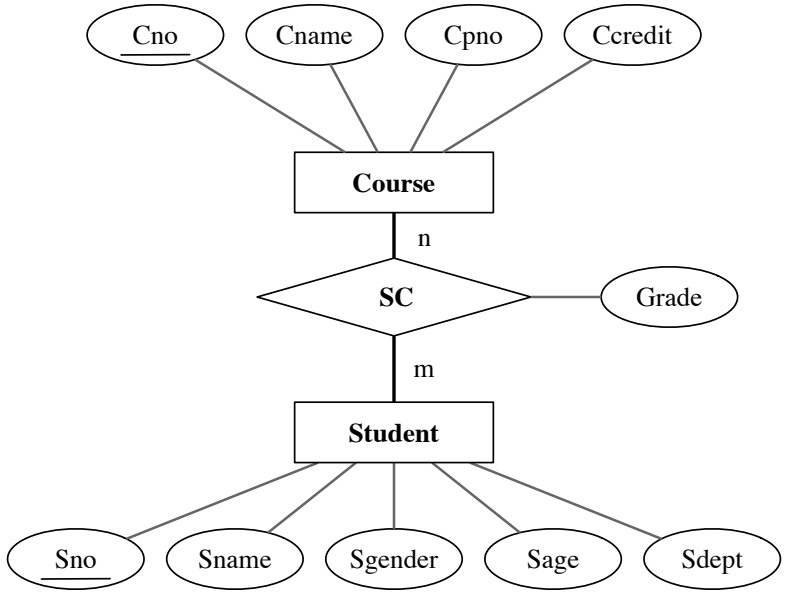
- **E-R图是用来描述实体集、属性和联系的图形化表示。**
  - **实体集**：用矩形表示，矩形框内标注实体集名。
  - **属性**：用椭圆形表示，并用无向边将其与相应的实体集连接起来。
  - 实体集的标识符用下划线标识出来。
    - ◆例如，Student实体集具有学号（Sno）、姓名（Sname）、性别（Sgender）、年龄（Sage）、系（Sdept）等属性，Sno是Student的标识符，E-R图表示如下



E-R图（续）

➢ E-R图

- **联系**：用菱形表示，菱形框内标注联系名，并用无向边分别与有关实体型连接起来，同时在无向边旁标上联系的类型（**1∶1，1∶n 或 m∶n**）。
- 联系可具有属性。如果实体之间的联系也具有属性，则用无向边连接属性和菱形。
  - ◆例如，联系SC有成绩Grade的属性，使用无向边将二者连接。



E-R模型

2 E-R模型

2.1 E-R模型的基本元素

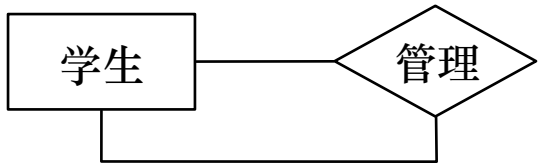
- 实体和实体集
- 属性
- 联系

2.2 E-R图

2.3 E-R联系类型

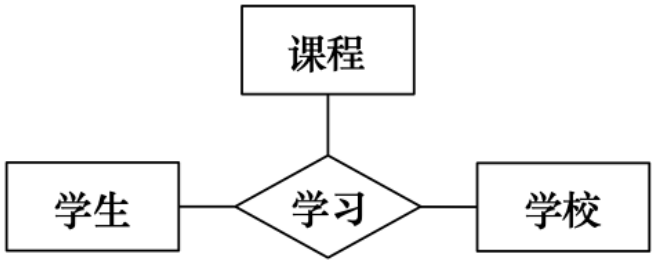
E-R联系类型

- **E-R联系类型是指实体之间不同关系的形式，常见的E-R关系：**
  - 一元联系
  - 二元联系
  - 三元联系

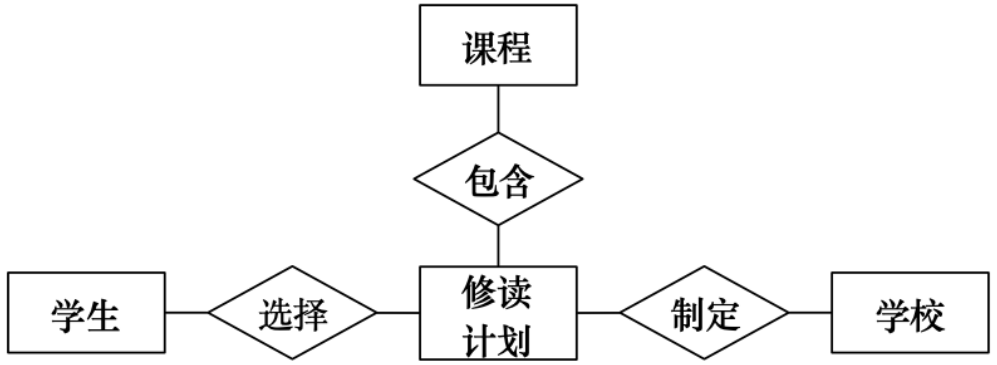


一元联系示例

二元联系示例



三元联系示例

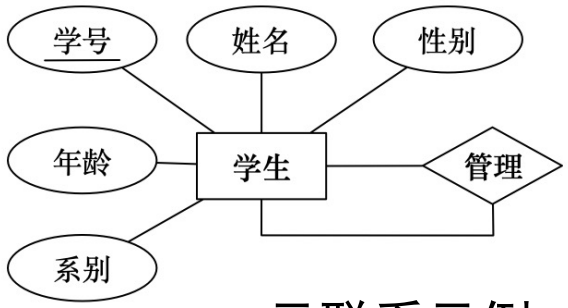


三元联系分解为二元联系示例

E-R联系类型（续）

➢ 一元联系

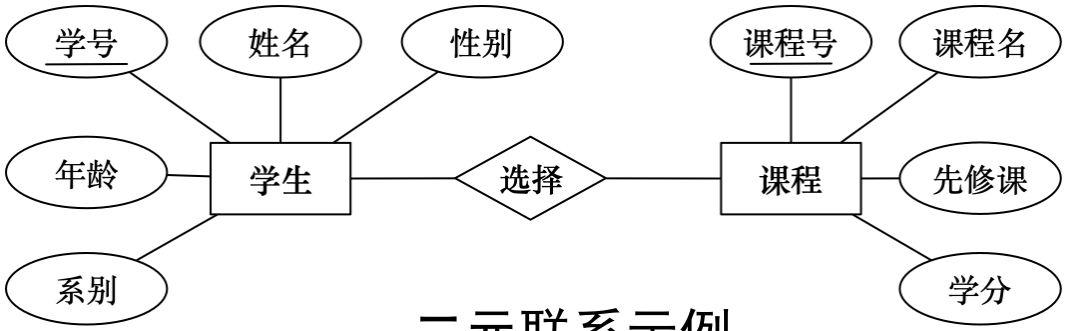
- 实体集内部实体之间的联系，即实体自己与自己之间的联系称为一元联系。
- 学生实体之间存在同级“管理”的关系，例如，学生群体中设立了班长一职，协助老师管理学生



一元联系示例

➢ 二元联系

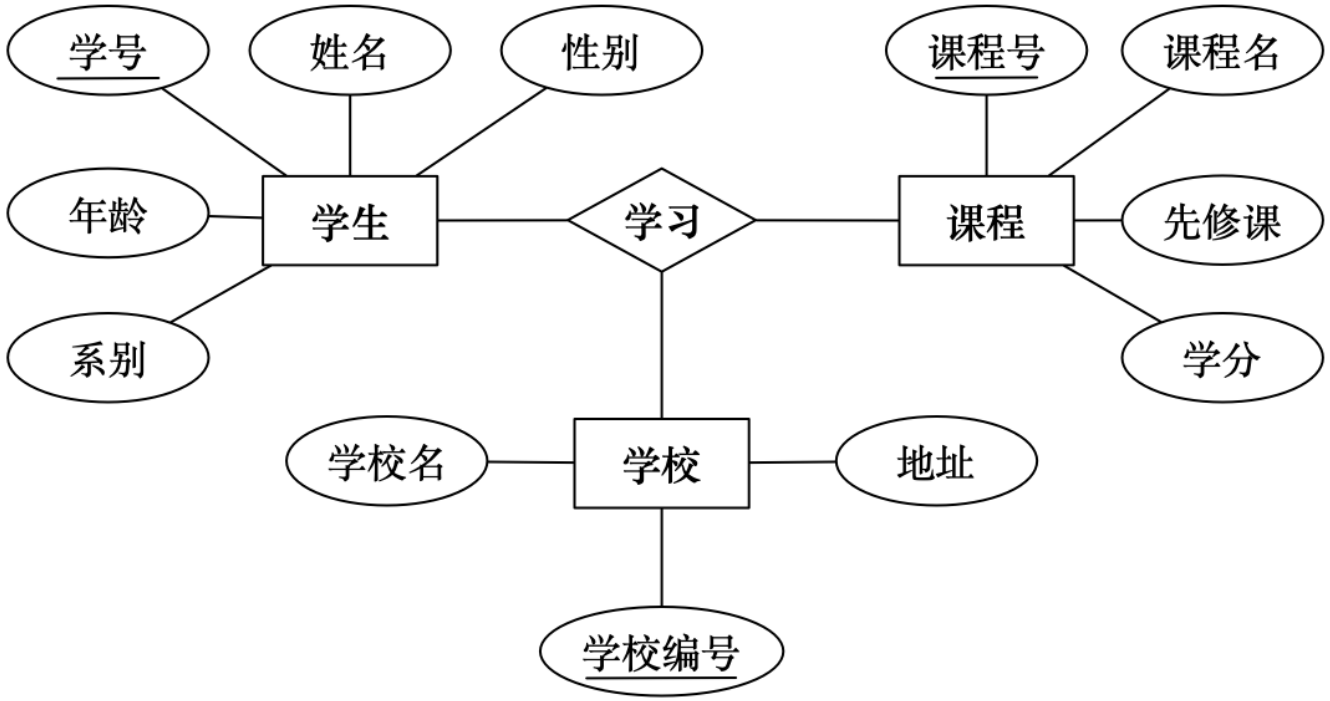
- 两个实体集之间的联系称为二元联系
- 一元联系可以看做是特殊的二元联系
- 例如，学生实体集与课程实体集之间存在“选择”课程的联系。



二元联系示例

E-R联系类型（续）

- **三元联系**
  - 三个实体集之间的联系称为三元联系
  - 例如，学生、学校和课程之间存在“学习”的联系，即某学生在某学校学习某课程

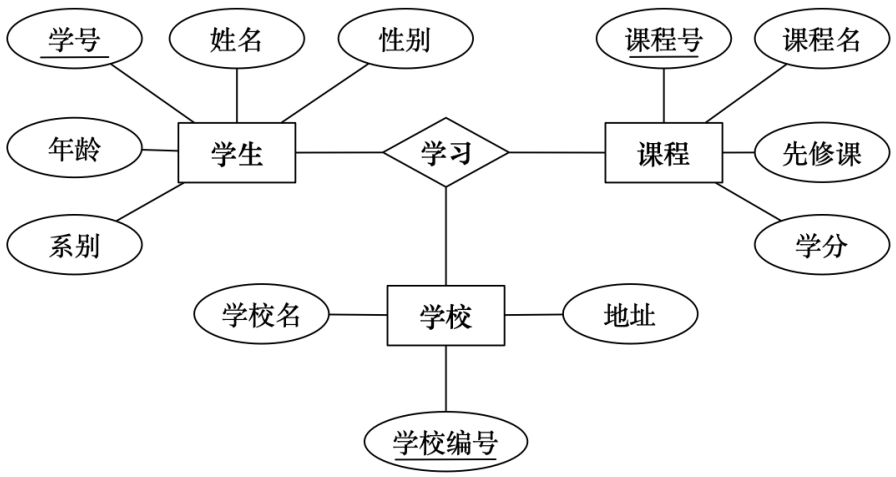


三元联系示例

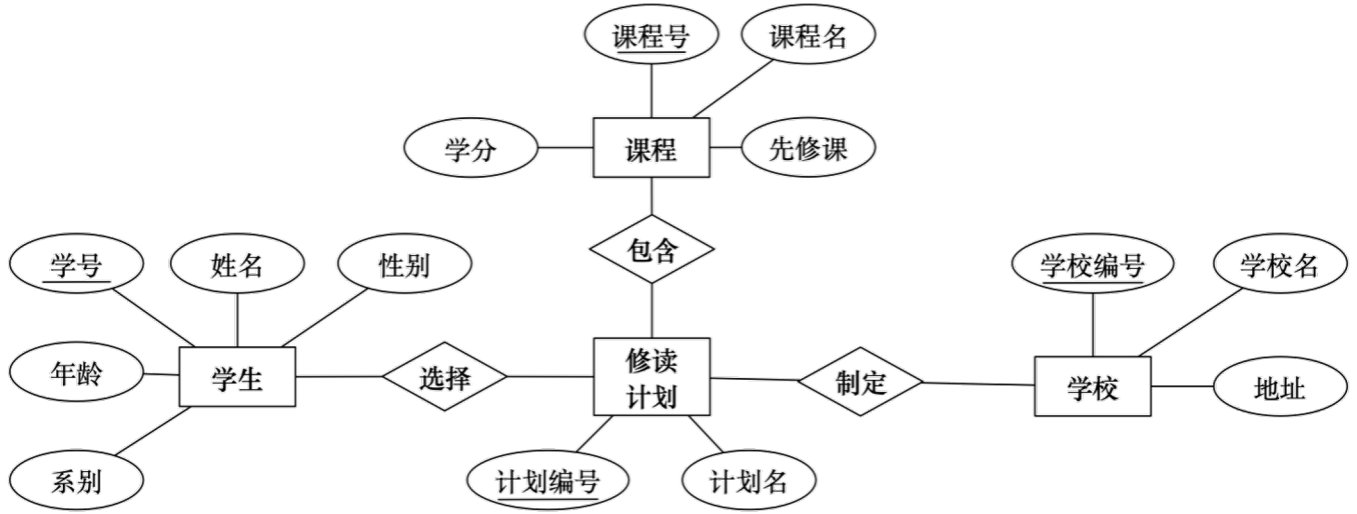
三元联系分解为二元联系

- 针对三元联系，可以将其分解为二元联系
- 考虑一个抽象的三元联系R，它将实体集A、B和C联系起来，将该三元联系转换成多个等价的二元联系的具体步骤如下：
  - （1）**用新实体集E替代联系R**。如果联系R有属性，则将这些属性赋给新建的实体集E；否则，为E建立一个特殊的标识性属性。因为每个实体集都应该至少有一个属性或多个属性的集合，以区别实体集中的各个成员；
  - （2）**建立三个新的联系R_A, R_B, R_C**。其中，R_A是实体集E和A之间的联系；R_B是实体集E和B之间的联系；R_C是实体集E和C之间的联系；
  - （3）**针对联系R中的每个联系(a_i, b_i, c_i)，在E中创建一个新实体e_i**，e_i代表(a_i, b_i, c_i)。然后，在三个新联系集中分别建立新的联系：在R_A中插入(e_i, a_i); 在R_B中插入(e_i, b_i); 在R_C中插入(e_i, c_i)。

三元联系分解为二元联系（续）



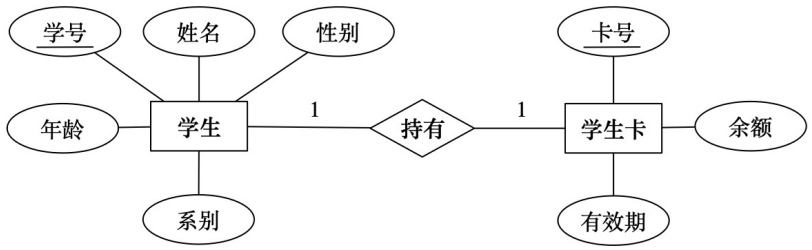
三元联系示例



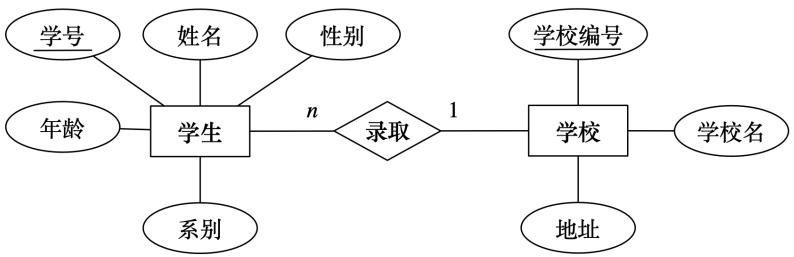
三元联系分解为二元联系示例

E-R联系类型（续）

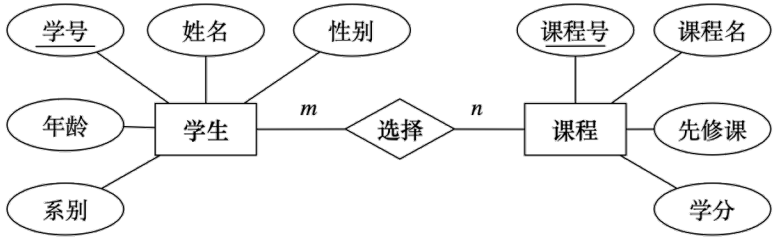
- 在实际应用中，二元联系是最常见的实体联系类型
- 对于二元联系，又能细分成以下几种类型
  - 一对一联系（1 : 1）
  - 一对多联系（1 : *n*）
  - 多对多联系（*m* : *n*）



一对一联系示例



一对多联系示例

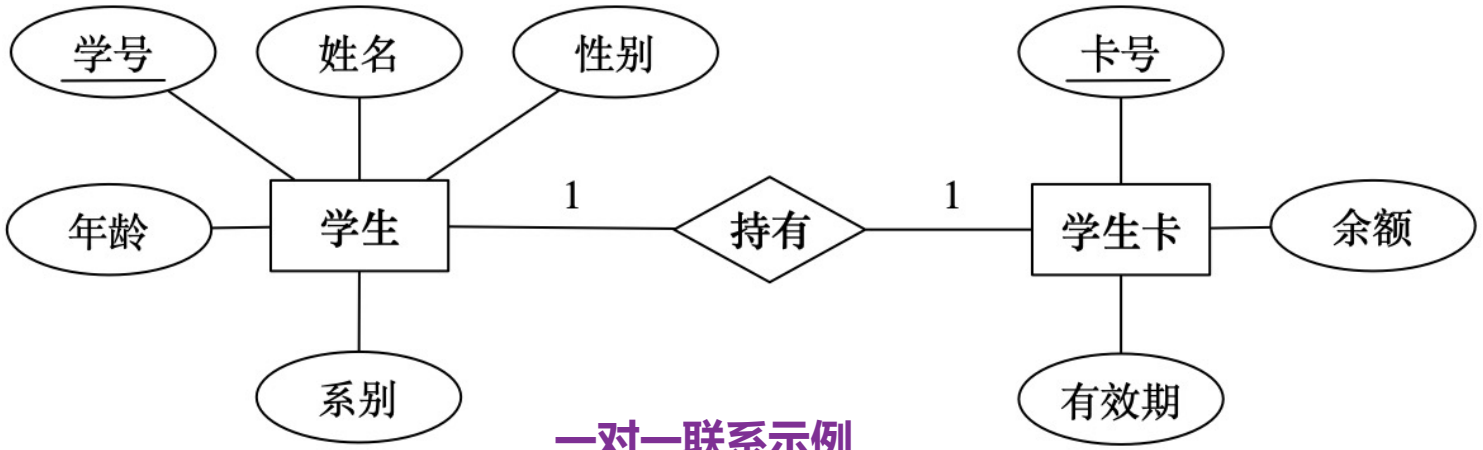


多对多联系示例

E-R联系类型（续）

➢ 一对一联系（1:1）

- 如果实体集E中每一个实体最多可以和另一实体集F中的一个实体有关系，反之亦然，则称实体集E和实体集F具有一对一联系，记作1:1。
- 例如，每个学生只有一张学生证，而一张学生证也只能对应一个学生，学生和学生证实体之间的联系即是“一对一”联系，如下图所示。

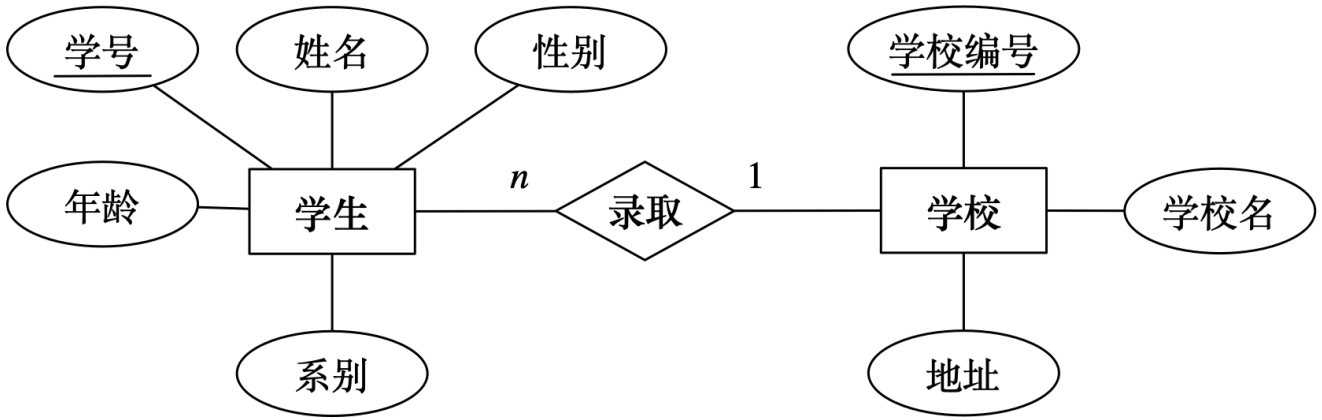


一对一联系示例

E-R联系类型（续）

➢ 一对多联系（1:*n*）

- 如果实体集E中的每一个实体，实体集F中有n个实体（n≥0）与之相关，则称实体集E与实体集F有一对多联系，记作1:n。
- 例如，假设一个学校学习可以录取多名学生，而每名学生只能被一个学校录取，则学校和学生之间建立起的这种“录取”联系就是一个“一对多”联系，如下图所示。

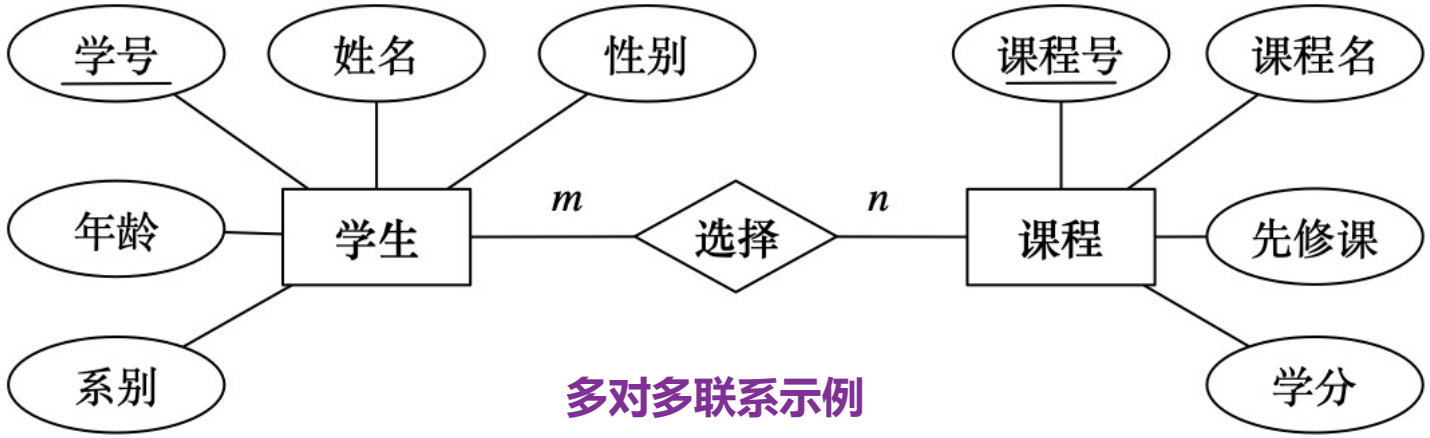


一对多联系示例

E-R联系类型（续）

➢ 多对多联系（m:n）

- 如果实体集E中的每个实体在实体集F中有n个实体（$n \geq 0$）与之相关，反之，实体集F中的每个实体都在实体集E中有m个实体（$m \geq 0$）与之相关，则称E和F具有多对多联系，记为$m:n$。
- 例如，一个学习选择多门课程，而一门课程也可以被多个学生选择，学生和课程实体集之间的“选择”联系就是一个“多对多”联系，如下图所示。



多对多联系示例

目录

E-R图转化为关系

➢ **转换步骤**

实体集的转换

- 实体集转换成一个关系模式（表头）
- 实体集的属性转换为关系模式的属性（表的列名）
- 实体集的标识符转换为关系模式的键

联系的转换

- 参照完整性约束，将E-R模型中的联系转化为关系表间的参照完整性约束

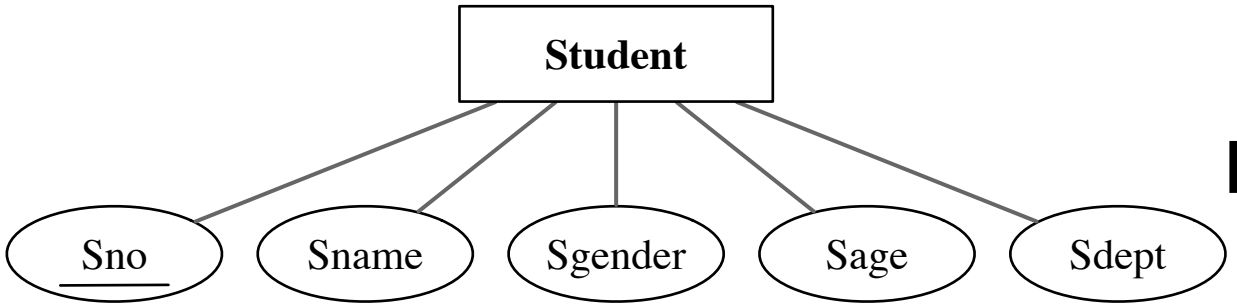
规范化设计

- 正确规范化所有关系表

E-R实体集的转换

➢ E-R实体集的转换

- ➢ 将E-R模型转换为关系模型的基本操作是将**实体集转换为关系表。**
- ➢ 关系表的表名是实体集的名称
- ➢ 关系表的属性是实体集的属性
- ➢ 关系表的主键是实体集的标识符



主键
Primary Key<PK>

Sno（学号）	Sname（姓名）	Sgender（性别）	Sage（年龄）	Sdept（所在系）
2021310721	李博	男	17	CS
2021310722	赵宇	男	19	CS
2021310723	张敏	女	18	CS
2021310724	王勇	男	18	MA
2021310725	刘佳	女	17	MA

E-R联系的转换

➢ E-R联系的转换

- 将E-R模型中的实体集转换成关系模式中的关系表后，还需要进一步将E-R模型中的联系转换成关系表之间的参照完整性约束
- **一元联系**：转换规则与二元联系类似
- **二元联系**：是E-R关系中最常见的关系类型，做重点介绍
- **三元联系**：可以将其先分解成二元联系，再用二元联系的规则进行转换

E-R联系的转换（续）

➢ E-R联系中二元联系的转换

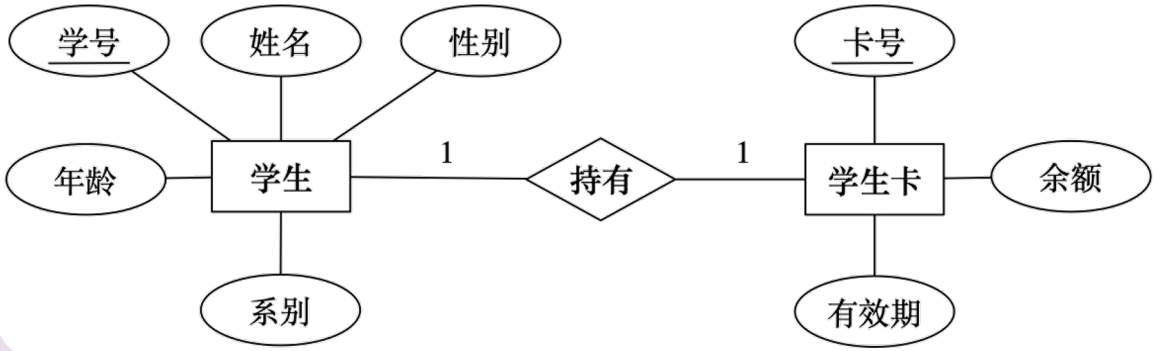
- **一对一（1:1）联系**的两种转换方式：
  - ◆（1）在两个实体集转换成的两个关系模式的基础上，在其中任意一个关系模式的属性中加入另一个关系模式的键和该联系自有的属性，这种转换不需要添加新的关系
  - ◆（2）在两个实体集转换成的两个关系模式的基础上，添加一个新关系模式，该关系模式的属性包含该联系相关的各实体集的标识符以及该联系自有的属性，其候选键可以是每个实体集的标识符

E-R联系的转换（续）

E-R联系中二元联系的转换

- **一对一（1:1）联系**的转换示例
- （1）在上述两个实体集对应的关系模式中任选一个添加另一个关系模式的主键即可
  - ◆即转化后的关系模式可为：
  - ◆Student(Sno, Sname, Sgender, Sage, Sdept, StuCardID)，StuCard(StuCardID, Cardbalance, CardExp)

  或
  - ◆Student(Sno, Sname, Sgender, Sage, Sdept), StuCard(StuCardID, Cardbalance, CardExp, Sno)
- （2）在两个实体集转换成的两个关系模式的基础上，添加一个新关系模式
  - ◆可添加一个独立的关系模式，即Hold(Sno, StuCardID)。
  - ◆此外，还有实体集转换生成的两个关系：
    - Student(Sno, Sname, Sgender, Sage, Sdept)
    - StuCard(StuCardID, Cardbalance, CardExp)



一对一联系示例

E-R联系的转换（续）

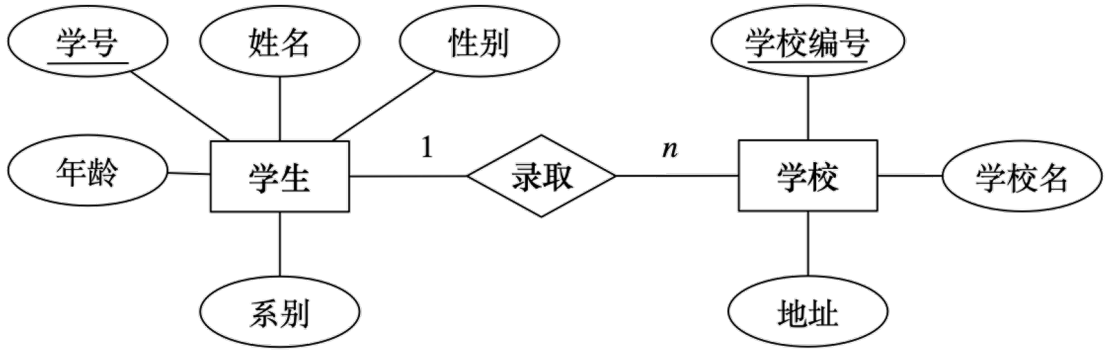
➢ E-R联系中二元联系的转换

- **一对多（1:n）联系**的两种转换方式：
- （1）在两个实体集转换成的两个关系模式的基础上，在n端实体类型转换成的关系模式中加入1端实体集的标识符和该联系自有的属性，这种转换不需要添加新的关系。
- （2）在两个实体集转换成的两个关系模式的基础上，为1:n联系添加一个新关系模式，该关系模式的属性是所有与该联系相关的各实体集的标识符以及该联系自有的属性，该关系模式的键是n端实体集的标识符。

E-R联系的转换（续）

E-R联系中二元联系的转换

- **一对多（1:n）联系**的转换示例：
- （1）在n端学生实体集Student转化成的关系模式中加入1端School实体集的标识符和该联系自有的属性，即转换后有：
  - ◆Student关系模式为Student(Sno, Sname, Sgender, Sage, Sdept, SchoolID)
  - ◆School关系为School(SchoolID, SchoolName, SchoolLocation)
- （2）添加一个独立的关系模式，该关系模式的键是n端实体集Student的标识符
  - ◆添加一个独立的关系模式，即 Admission(Sno, SchoolID)。
  - ◆此外，结合实体集转换生成的两个关系：
    - Student(Sno, Sname, Sgender, Sage, Sdept)
    - School(SchoolID, SchoolName, SchoolLocation)

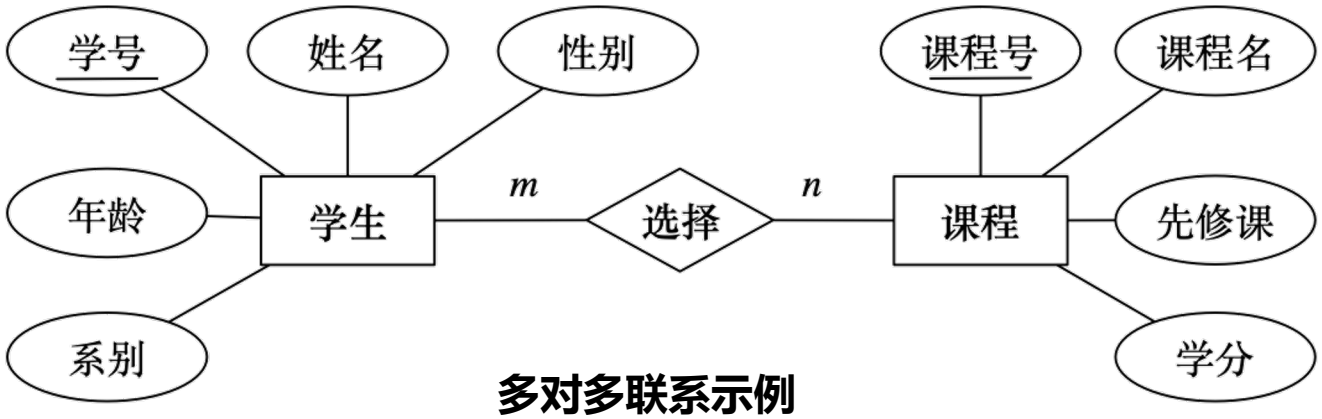


一对多联系示例

E-R联系的转换（续）

E-R联系中二元联系的转换

- **多对多（m:n）联系**：对该联系添加一个新的关系模式，这个新的关系模式的属性是两端实体类型的标识符以及该联系自有的属性，其键为两端实体集的标识符的组合。
- 学生实体集Student和课程实体集Course的“选择”联系SC是二元联系中的m:n联系，可以新建一个新的关系SC，并将与SC相关的各实体集的键以及SC联系自有的属性（如Grade）转为新关系的属性，其键为两端实体集的标识符的组合，即SC联系转换的关系为：SC(Sno, Cno, Grade)

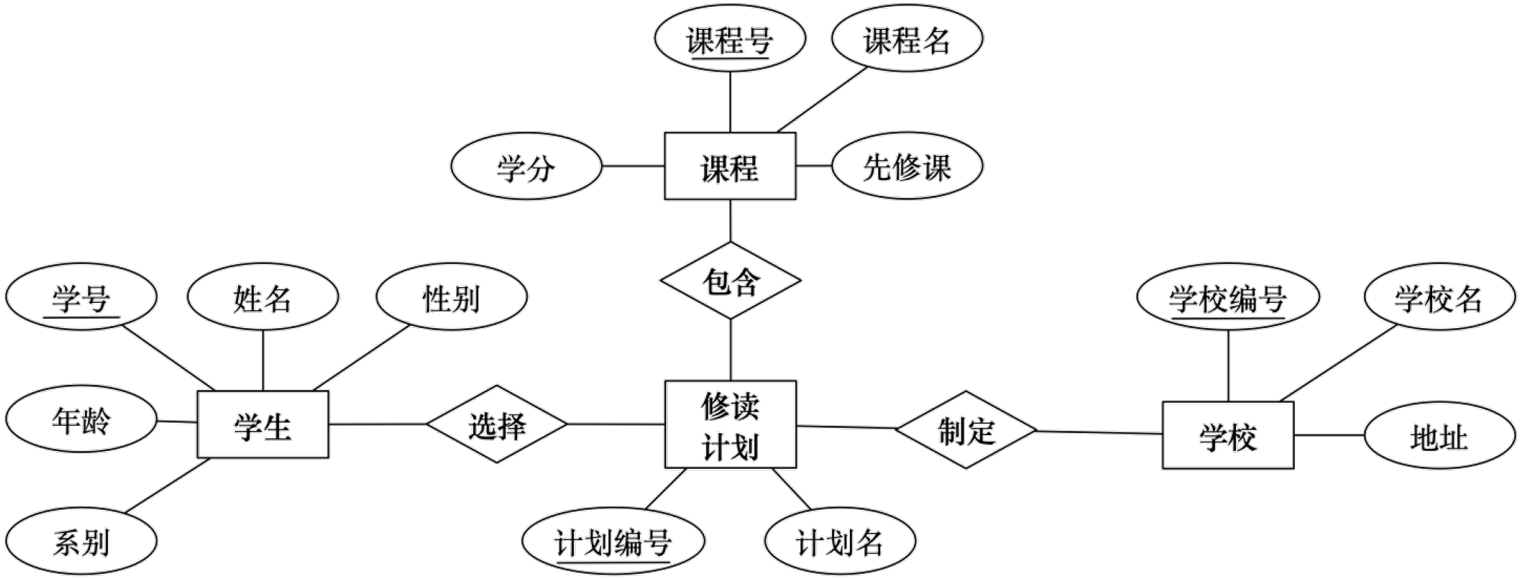


多对多联系示例

SC表

Sno（学号）	Cno（课程号）	Grade（成绩）
2021310721	5	98
2021310722	1	87
2021310723	1	92
2021310723	5	76
2021310724	7	84
2021310725	4	95

E-R图转换为关系示例



- 学生关系　Student(Sno, Sname, Sgender, Sage, Sdept)
- 课程关系　Course(Cno, Cname, Cpno, Ccredit)
- 学校关系　School(SchoolID, SchoolName, SchoolLocation)
- 修读计划关系　Plan(PlanID, PlanName)
- 包含关系　Contain(PlanID, Cno)
- 制定关系　MakePlan(PlanID, SchoolID)
- 选择关系　SelectPlan(PlanID, Sno)

目录

为什么要规范化设计?

- **数据库规范化设计**是指在数据库中**减少数据冗余和定义一个规范的表间结构**，以更好地**实现数据完整性和一致性**。
  - 数据冗余是指一组数据不必要地重复出现在多个表中。
  - 冗余数据会占用更多存储空间并且提高数据库的维护成本，例如维护数据一致性。
- 数据库规范化设计具有以下几个优点：

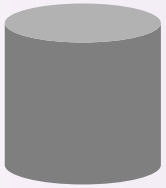
（1）降低数据存储和维护数据一致性的成本。即减少冗余数据，减少存储空间的开销；当数据发生更新（以及插入和删除）的时候，不必同时更新那些冗余数据，降低了维护数据一致性的成本。

（2）便于设计合理的关系表间的依赖和约束关系，便于实现数据完整性和一致性，避免插入异常、删除异常、更新异常。

（3）便于设计合理的数据库结构，以提高数据库系统的整体性能。

重要概念之间的关系

关系数据库概念



关系数据库基本概念

- 关系表、属性、元组、分量
- 关系模式、关系示例

关系数据库设计

概念设计阶段

E-R模型

数据库设计的高阶表示模型

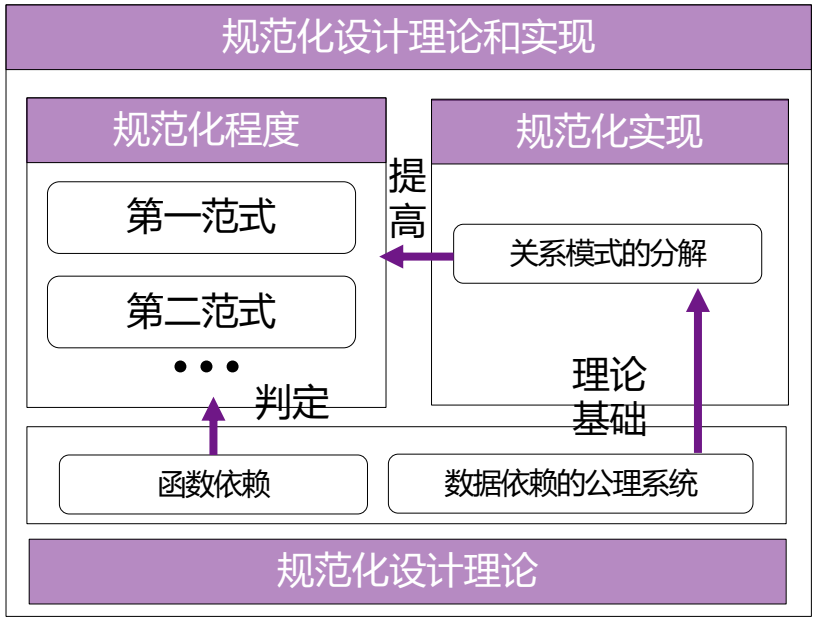
从E-R图到关系设计

逻辑设计阶段

关系模型

规范化设计

(关系数据库的数据模型)



数据库规范化设计

4、数据库规范化设计理论

4.1 函数依赖理论

4.2 规范化设计和范式

4.3 数据依赖的公理系统

数据库规范化设计

- 为了解决关系模式设计的规范化问题，需要引入一个重要概念：**函数依赖**（functional dependency）
- **函数依赖**反映了一个关系中属性或者属性组之间相互依存、相互制约的关系，即两个列或者列组之间的约束。基于函数依赖理论，可以将一个关系分解为几个更小的关系，使之满足规范化程度更高的关系表。
- **函数依赖**是关系中属性之间在语义上的关联特性。例如，假设一个关系表的A列是国家，B列是首都，那么如果两条记录的A列值是一样的，那么必然它们B列的值也是一样的，即国家决定了首都。

函数依赖

- 在关系模式设计上最重要的约束是函数依赖，它反映了一个关系表中列或者列组之间相互依存、相互制约的关系，即两个列或者列组之间的约束。

【例3.1】描述一个学生Student关系，可以有学号、姓名、系名等属性。一个学号只对应一个学生，一个学生只在一个系中学习，“学号”值确定后，学生的姓名及所在系的值就被唯一确定。

- 有如下依赖关系，Sname=f(Sno)，Sdept=f(Sno)
  - 即Sno函数决定Sname
  - Sno函数决定Sdept
  - 记作Sno→Sname，Sno→Sdept

函数依赖（续）

- 函数依赖定义
  - 对一关系$R(U)$，X和Y是其列集合U的子集，t和l分别是R中的任意两个元组。
  - 如果$t[X]=l[X]$，则$t[Y]=l[Y]$，那么称Y函数依赖于X，或者X函数决定Y，记为$X \to Y$。
  - 如果Y不依赖于X，则记为$X \nrightarrow Y$。如果$X \to Y$且$Y \to X$，则X与Y一一对应，记为$X \leftrightarrow Y$。
- 一个函数依赖要成立，不但要求关系R中当前的值都能满足函数依赖条件，而且还要求关系中的任一可能取值都满足函数依赖的条件。此外，函数依赖还具有数据语义特征，即函数依赖在某种程度上也是现实世界的反映。

函数依赖（续）

➢ 函数依赖举例

【例3.2】对于关系Student(Sno, Sname, Sgender, Sage, Sdept)，假设不允许重名，则有如下函数依赖:

- Sno → Sgender，Sno → Sage
- Sno → Sdept，Sno ⟷ Sname
- Sname → Sgender，Sname → Sage
- Sname → Sdept
- 但 Sgender ↛ Sage, Sgender ↛ Sdept

Student关系

Sno（学号）	Sname（姓名）	Sgender（性别）	Sage（年龄）	Sdept（所在系）
2021310721	李博	男	17	CS
2021310722	赵宇	男	19	CS
2021310723	张敏	女	18	CS
2021310724	王勇	男	18	MA
2021310725	刘佳	女	17	MA

函数依赖（续）

- 函数依赖举例
  - 对于关系Student(Sno, Sname, Sgender, Sage, Sdept)

违背了Sno → Sname

Student关系

Sno（学号）	Sname（姓名）	Sgender（性别）	Sage（年龄）	Sdept（所在系）
2021310721	李博	男	17	CS
2021310721	赵宇	男	19	CS
2021310723	张敏	女	18	CS
2021310724	王勇	男	18	MA
2021310725	刘佳	女	17	MA

函数依赖（续）

➢ 函数依赖举例

- 通过下面Student关系实例，是否能得出Sname → Sage ？
  - ◆ “Sname → Sage ” 这个函数依赖只有在不允许有同名人的条件下成立

Student关系

Sno（学号）	Sname（姓名）	Sgender（性别）	Sage（年龄）	Sdept（所在系）
2021310721	李博	男	17	CS
2021310722	赵宇	男	19	CS
2021310723	张敏	女	18	CS
2021310724	王勇	男	18	MA
2021310725	刘佳	女	17	MA

函数依赖不是指关系模式R的某个或某些关系实例满足的约束条件，而是指R的所有关系实例均要满足的约束条件。

函数依赖（续）

- 平凡函数依赖和非平凡函数依赖
- 完全函数依赖和部分函数依赖
- 传递函数依赖
- 多值依赖
- 关系的键

平凡函数依赖和非平凡函数依赖

- $X \rightarrow Y$，但$Y \not\subseteq X$,则称$X \rightarrow Y$是**非平凡函数依赖**
- $X \rightarrow Y$，但$Y \subseteq X$,则称$X \rightarrow Y$是**平凡的函数依赖**
- 若$X \rightarrow Y$，则X称为这个函数依赖的**决定因素**
- 若$X \rightarrow Y$，$Y \rightarrow X$，则记作$X \leftrightarrow Y$
- 若Y不函数依赖于X，则记作$X \nrightarrow Y$

非平凡的函数依赖

Sno → Sgender,Sno → Sname

Sno → Sage, Sno → Sdept,…

平凡函数依赖： Sno → Sno，…

Student关系

Sno（学号）	Sname（姓名）	Sgender（性别）	Sage（年龄）	Sdept（所在系）
2021310721	李博	男	17	CS
2021310722	赵宇	男	19	CS
2021310723	张敏	女	18	CS
2021310724	王勇	男	18	MA
2021310725	刘佳	女	17	MA

所有关系都满足平凡函数依赖，平常所指的函数依赖一般都是指非平凡函数依赖！

完全函数依赖和部分函数依赖

- 假设X和Y是关系$R(U)$属性集U的不同子集，如有$X \to Y$且不存在X的真子集X'，使得$X' \to Y$，则有$X \xrightarrow{f} Y$，称Y**完全函数依赖**于X；否则，记为$X \xrightarrow{p} Y$，称作Y**部分函数依赖**于X。
- 【**例3.3**】在关系SC(Sno, Cno, Grade)中，有：Sno↛Grade, Cno↛Grade,
  - 那么（Sno,Cno）$\xrightarrow{f}$ Grade，（Sno,Cno）$\xrightarrow{p}$ Sno，（Sno,Cno）$\xrightarrow{p}$ Cno

SC表

Sno （学号）	Cno （课程号）	Grade （成绩）
2021310721	5	98
2021310722	1	87

传递函数依赖

- 如果X、Y和Z分别是关系R(U)不同的列属性集合，如果存在$X \to Y$，$Y \not\to X$，$Y \to Z$，则称Z对X有**传递函数依赖**。
- 值得注意的是，在上述的定义中，我们强调了$Y \not\to X$，说明不存在$X \leftrightarrow Y$的情况，否则，Z就直接函数依赖于X，而不是传递函数依赖于X了。

【例3.4】在关系Course(Cno, Cname, Cpno，Ccredit)中，如果课程名可以相同，但课程号必须不同，且同一课程名的先行课是同样的。

那么有：Cno → Cname, Cname ↛ Cno, Cname → Cpno

则称： Cpno 对 Cno有传递函数依赖

多值依赖

- 对于关系R的属性集U，设X和Y是U的子集，$Z = U - X - Y$，令(x, y, z)表示属性集(X, Y, Z)的元组。如果存在(x, y_1, z_1)和(x, y_2, z_2)时，也存在(x, y_1, z_2)和(x, y_2, z_1)，那么称**多值函数依赖**$X \rightarrow\rightarrow Y$在关系$R$上成立。

关系模式$R(U)$中多值依赖（multivalued dependency, MVD）$X \rightarrow\rightarrow Y$成立，当且仅当对$R(U)$的任一关系$r$，给定一对$(x, z)$值，有一组$Y$的值，这组值仅仅决定于$x$值而与$z$值无关。直观的说，对于任意的元组$a, b$且$a[X] = b[X]$，交换元组$a, b$的$Y$属性的值，交换后的元组仍存在于关系$R(U)$中，则$X \rightarrow\rightarrow Y$。

Sno （学号）	Phone （手机号）	Email （邮箱）
2021310721	137-XXXX-2271	boli@tsinghua.edu.cn
2021310721	137-XXXX-2271	libo@example.com
2021310721	158-XXXX-1790	boli@tsinghua.edu.cn
2021310721	158-XXXX-1790	libo@example.com
2021310722	199-XXXX-2290	yuzhao@tsinghua.edu.cn
2021310722	199-XXXX-2290	zhaoyu1999@exmaple.com

多值依赖

- 给定关系模式$R(U)$，X,Y,Z是U的子集，并且$Z=U-X-Y$。如果$X\rightarrow\rightarrow Y$成立，且Z为空集，则$X\rightarrow\rightarrow Y$是平凡多值依赖；如果Z非空，则$X\rightarrow\rightarrow Y$是非平凡多值依赖。
- 函数依赖其实是多值依赖的一个特例，函数依赖其实是单值依赖，即如果Y组的值仅有一个，则多值依赖就成为了函数依赖。

连接依赖

- 连接依赖（Join Dependency）的定义如下：如果一个关系可以被分解为若干子关系，并且对这若干子关系通过连接（Join）操作后得到原始关系，则连接依赖成立。
- **多值依赖是连接依赖的特例**，即多值依赖可以分解成两个子关系的连接，而连接依赖则可以是多个（包含2个）子关系的连接。

Sno （学号）	Phone （手机号）	Email （邮箱）	Bank （银行卡）
2021310721	137-XXXX-2271	boli@tsinghua.edu.cn	110
2021310721	137-XXXX-2271	libo@example.com	110
2021310721	158-XXXX-1790	boli@tsinghua.edu.cn	110
2021310721	158-XXXX-1790	libo@example.com	110
2021310722	199-XXXX-2290	yuzhao@tsinghua.edu.cn	111
2021310722	199-XXXX-2290	zhaoyu1999@exmaple.com	111

关系的键

- 使用函数依赖的方式来重新解释（定义）关系的键。
  - 如果一个属性或者多个属性的集合$K=\{A_1, A_2, \ldots, A_n\}$满足下列性质，则认为$K$是关系$R(U)$的**候选键**（candidate key）。
    - ◆其他属性函数依赖于该属性或属性集;
    - ◆其他属性都不函数依赖于该属性或属性集的任一真子集。
  - 上述两个规则就是U完全函数依赖于K，即$K \xrightarrow{f} R$。
  - 如果一个关系中有多个候选键，则选择其最重要的一个候选键作为关系$R(U)$的**主键**（primary key）。

关系的键（续）

【例3.5】考虑SC关系，其键为{Sno, Cno}，即 $(\mathrm{Sno}, Cno) \xrightarrow{f} Grade$

- 首先要证明它们函数决定了其他所有属性，即如果{Sno, Cno}组合的取值相同，则在其他属性Grade上的取值也应该相同。
- 根据上面讨论的性质，{Sno, Cno}的任意真子集都不能函数决定其他的属性
- 因此，{Sno, Cno}是关系SC的键。

SC表

Sno （学号）	Cno （课程号）	Grade （成绩）
2021310721	5	98
2021310722	1	87
2021310723	1	92
2021310723	5	76
2021310724	7	84
2021310725	4	95

数据库规范化设计

4、数据库规范化设计理论

4.1 函数依赖理论

4.2 规范化设计和范式

4.3 数据依赖的公理系统

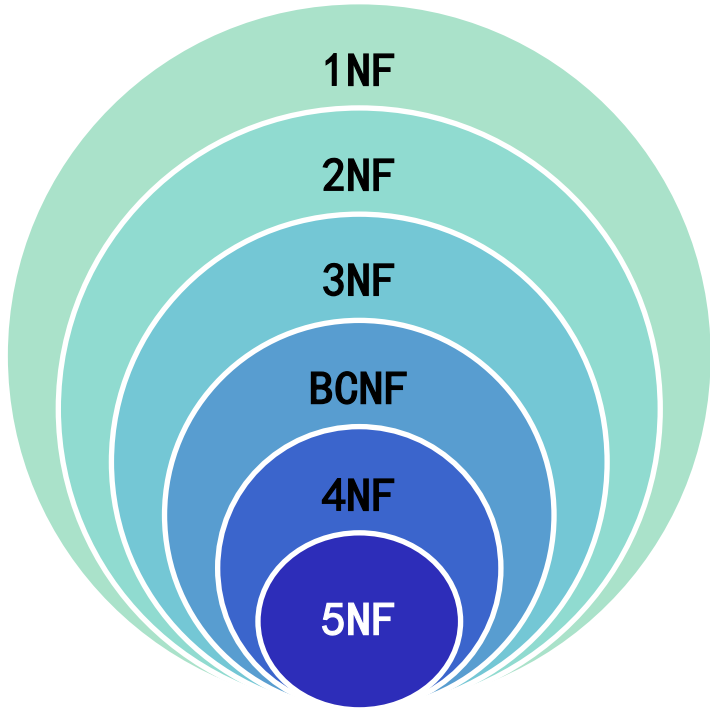
规范化设计和范式

- **规范化设计**：在关系模式中存在函数依赖时就有可能存在数据冗余，进而可能导致数据操作异常。因此，关系表的规范化设计就是要尽可能地减少关系表中列或者列组之间的依赖关系，进而得到简洁独立的关系表。

- **范式**
  - 【**定义1**】关系表的规范程度状态为**范式**（normal form, NF）
  - 【**定义2**】范式是符合某一种级别的关系模式的集合
    - ◆范式可以用于确保数据库模式中没有各种类型的异常和不一致，不同的规范化范式要求可以设计出冗余程度不同的数据库。

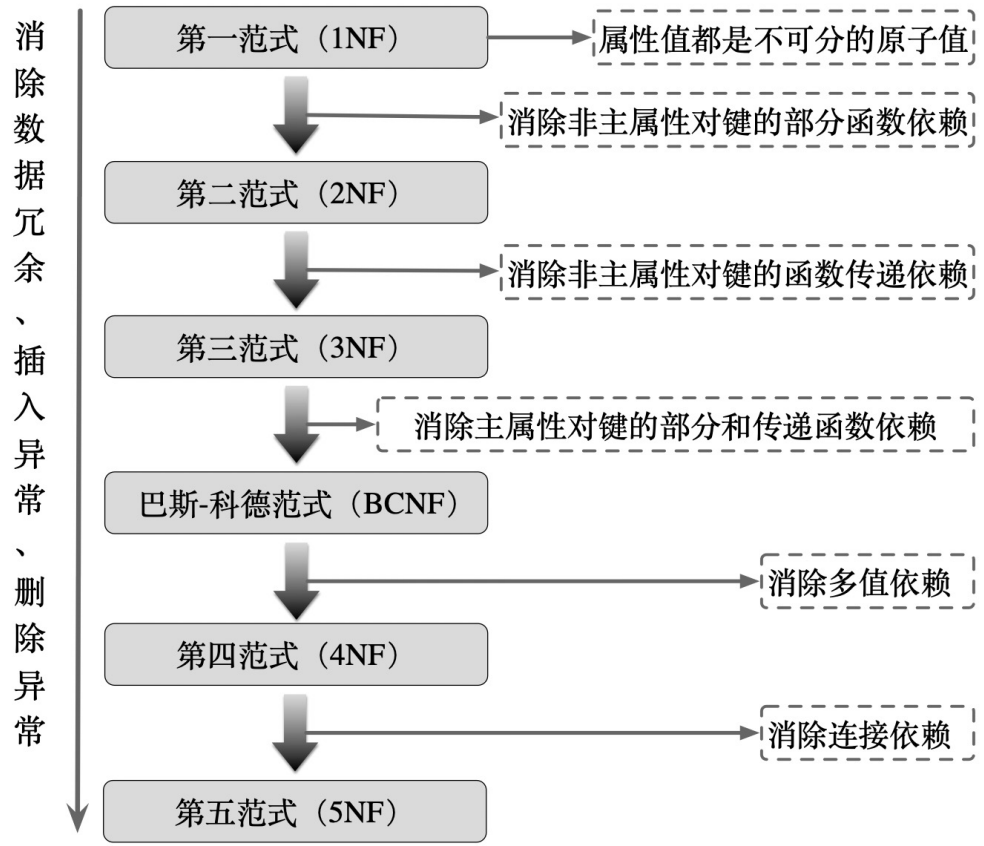
范式

- **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。**
- **常见的范式**
  - 第一范式（First Normal Form, 1NF）
  - 第二范式（Second Normal Form, 2NF）
  - 第三范式（Third Normal Form, 3NF）
  - 巴斯-科德范式（Boyce-Codd Normal Form, BCNF）
  - 第四范式（Fourth Normal Form，4NF）
  - 第五范式（Fifth Normal Form, 5NF）



范式和规范化

- 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**(normalization)。



规范化过程和各范式示意图

第一范式 1NF

- **第一范式**（1NF）是指关系*R*的每一属性都是不可再分的基本数据项，同一属性中不能有多个值，即关系表中的某个属性不能有多个值或者不能有重复的属性。
- 在关系数据库中，满足最低要求的范式是第一范式，不满足第一范式的不是关系数据库。如果出现重复的属性，则根据第一范式，需要将该属性进行细分。

Sno （学号）	Sname （姓名）	Contact （联系方式）
2021310721	李博	137-XXXX-1790，Lib@uni.edu.cn
2021310722	赵宇	130-XXXX-1110，Zhaoy@uni.edu.cn
2021310723	张敏	139-XXXX-9999，Zhangm@uni.edu.cn

原始的Student关系表

Sno （学号）	Sname （姓名）	Phone （手机号）	Email （邮箱）
2021310721	李博	137-XXXX-1790	Lib@uni.edu.cn
2021310722	赵宇	130-XXXX-1110	Zhaoy@uni.edu.cn
2021310723	张敏	139-XXXX-9999	Zhangm@uni.edu.cn

满足1NF的Student关系表

第二范式 2NF

【例3.6】考虑下图的学生-选课-住址关系SLC，其复合主键是{Sno, Cno}，Sloc为学生的宿舍楼号，并且每个系的学生住在同一个地方。SLC关系存在如下函数依赖：$(Sno, Cno) \xrightarrow{f} Grade, Sno \rightarrow Sdept$，$(Sno, Cno) \xrightarrow{p} Sdept$，$Sno \rightarrow Sloc$，$(Sno, Cno) \xrightarrow{p} Sloc$，$Sdept \rightarrow Sloc$。

学生宿舍地址Sloc和系别Sdept只依赖于复合键中Sno，存在部分函数依赖。

Sno （学号）	**Sdept** （所在系）	**Sloc** (宿舍)	**Cno** （选课号）	**Grade** （成绩）
2021310721	CS	ZJ#15	CS204	90
2021310721	CS	ZJ#15	CS207	88
2021310722	MA	ZJ#15	MA192	79
2021310723	CS	ZJ#15	CS207	92

满足1NF的SLC表

非主属性Sdept、Sloc并不完全依赖于主键

第二范式 2NF（续）

- 满足1NF的SLC关系，会存在以下问题
  - 插入异常
    - ◆假设新入学一批学生，还没有选课，然而由于课程号Cno是主属性，则无法将这批学生插入
  - 删除异常
    - ◆如果学号Sno为2021310721的学生只选了一门课CS204，现在他不再选这门课，则删除CS204后，整个元组的其他信息（所在系和宿舍信息）也被删除了。
  - 修改复杂
  - 存储冗余
    - ◆ 如果一个学生选了多门课，则Sdept，Sloc被重复存储多次。如果该生转系，则需要修改所有关联的Sdept和Sloc，造成修改的复杂化。

Sno（学号）	Sdept（所在系）	Sloc(宿舍)	Cno（选课号）	Grade（成绩）
2021310721	CS	ZJ#15	CS204	90
2021310721	CS	ZJ#15	CS207	88
2021310722	MA	ZJ#15	MA192	79
2021310723	CS	ZJ#15	CS207	92

满足1NF的SLC表

第二范式 2NF（续）

- 出现上述异常的原因
  - 关系SLC存在部分函数依赖，有两类非主属性
    1. 一类如Grade，它对键完全函数依赖
    2. 另一类如Sdept、Sloc，它们对键不是完全函数依赖
- 为了解决上述问题，可以通过分解关系SLC进行消除部分函数依赖。
- 由此，引出了第二范式的概念。

Sno （学号）	Sdept （所在系）	Sloc (宿舍)	Cno （选课号）	Grade （成绩）
2021310721	CS	ZJ#15	CS204	90
2021310721	CS	ZJ#15	CS207	88
2021310722	MA	ZJ#15	MA192	79
2021310723	CS	ZJ#15	CS207	92

满足1NF的SLC表

第二范式 2NF（续）

- **第二范式**（2NF）是指关系*R*首先要满足第一范式，并且每一个非主属性都完全函数依赖于任何一个候选键。

【**例3.6-续**】由此，我们将关系模式SLC分解成两个满足2NF的关系模式，以消除上述的异常。

Sno（学号）	Sdept（所在系）	Sloc (宿舍)	Cno（选课号）	Grade（成绩）
2021310721	CS	ZJ#15	CS204	90
2021310721	CS	ZJ#15	CS207	88
2021310722	MA	ZJ#15	MA192	79
2021310723	CS	ZJ#15	CS207	92

满足1NF的SLC表

Sno（学号）	Sdept（所在系）	Sloc (宿舍)
2021310721	CS	ZJ#15
2021310722	MA	ZJ#15
2021310723	CS	ZJ#15

满足2NF的SL表

Sno（学号）	Cno（选课号）	Grade（成绩）
2021310721	CS204	90
2021310721	CS207	88
2021310722	MA192	79
2021310723	CS207	92

满足2NF的SC表

第三范式 3NF

【例3.7】考虑下图所示的SL关系表，该关系满足第二范式，但存在如下函数依赖：

- Sno → Sdept , Sdept → Sloc
- Sdept ↛ Sno, Sloc ↛ Sdept
- 因此存在传递函数依赖：Sno → Sloc。

Sno （学号）	**Sdept** （所在系）	**Sloc** (宿舍)
2021310721	CS	ZJ#15
2021310722	MA	ZJ#15
2021310723	CS	ZJ#15

满足2NF的SL表

第三范式 3NF（续）

- 满足2NF的SL关系，依然存在以下问题
  - 插入异常
    - ◆如果某个系刚成立但还没新生入学，则无法向数据库中插入该系的系名和学生宿舍地址信息。
  - 删除异常
    - ◆如果该系的所有学生都毕业了，在删除这些学生的时候，也会同时将该系的系名和学生宿舍地址信息。
  - 修改复杂
  - 存储冗余
    - ◆ 一方面同系别的学生宿舍住址信息重复存储；
    - ◆ 另一方面如果添加新的学生信息，还需要重复添加他们的住址信息 。

Sno（学号）	Sdept（所在系）	Sloc(宿舍)
2021310721	CS	ZJ#15
2021310722	MA	ZJ#15
2021310723	CS	ZJ#15

满足2NF的SL表

第三范式 3NF（续）

- 出现上述异常的原因
  - 关系SL存在传递函数依赖： Sno → Sloc
- 为了解决该问题，可以通过分解关系SL进行消除传递函数依赖。
- 由此，引出了第三范式的概念。

Sno （学号）	Sdept （所在系）	Sloc (宿舍)
2021310721	CS	ZJ#15
2021310722	MA	ZJ#15
2021310723	CS	ZJ#15

满足2NF的SL表

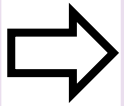
第三范式 3NF（续）

- **第三范式**（3NF）是指在关系R满足第一范式，并且不存在非主属性对候选键的传递函数依赖

【例3.7-续】将关系模式SL分解成两个关系SL和DL以消除传递函数依赖，使之满足第三范式。

Sno（学号）	Sdept（所在系）	Sloc (宿舍)
2021310721	CS	ZJ#15
2021310722	MA	ZJ#15
2021310723	CS	ZJ#15

满足2NF的SL表



Sno（学号）	Sdept（所在系）
2021310721	CS
2021310722	MA
2021310723	CS

满足3NF的SL表

Sdept（所在系）	Sloc (宿舍)
CS	CS_A107
MA	MA_B22
CS	CS_B23

满足3N的DL表

$$3NF \subset 2NF$$

巴斯-科德范式 BCNF

【例3.8】考虑下图所示的SCT关系，假设每位教师只教一门课程，每门课程可以有若干教师讲授，那么每位学生选修某门课程就对应一位教师。

- 关系SCT的函数依赖：
  - $(\text{Sno}, \text{Cno}) \rightarrow \text{Tno}$, $(\text{Sno}, \text{Tno}) \rightarrow \text{Cno}$, $\text{Tno} \rightarrow \text{Cno}$;
  - 它的候选键是：(Sno, Cno)和(Sno, Tno)
  - 主属性是Sno, Cno, Tno
  - 因为没有任何**非主属性**对键传递函数依赖或部分函数依赖，所以关系SCT是满足第三范式

Sno （学号）	Cno （选课号）	Tno （教师号）
2021310721	CS204	李老师 001
2021310722	MA192	马老师 001
2021310723	CS204	周老师 001

满足3NF的SCT关系表

巴斯-科德范式 BCNF（续）

➢ 满足3NF的SCT关系，依然存在以下问题

- 插入异常
  - ◆存在Tno → Cno函数依赖关系，如果某位教师开设的课程没有学生选修，无法将相关信息插入到数据库中
- 删除异常
  - ◆如果选修CS204课程的学生都毕业了，那么在删除这些学生信息时，相应的课程和教师信息也都同时被删掉了，造成了信息丢失。
- 修改复杂
- 数据冗余
  - ◆ 虽然一位教师只教授一门课程，但每位选修该教师的课程的学生元组都需要记录该信息，造成了重复存储。
  - ◆ 如果李老师001教授的CS204课程升级改版成CS207课程后，则需要对所有选修的该课程的元组进行修改 。

Sno（学号）	Cno（选课号）	Tno（教师号）
2021310721	CS204	李老师 001
2021310722	MA192	马老师 001
2021310723	CS204	周老师 001

满足3NF的SCT关系表

巴斯-科德范式 BCNF（续）

- 出现上述异常的原因
  - 关系SCT存在Tno → Cno
  - 即主属性Cno 部分函数依赖于候选键(Sno，Tno)
- 为了解决该问题，可以通过分解关系SCT进行**消除主属性对键的部分函数依赖。**
- 由此，引出了巴斯-科德范式 的概念。

Sno （学号）	**Cno** （选课号）	**Tno** （教师号）
2021310721	CS204	李老师 001
2021310722	MA192	马老师 001
2021310723	CS204	周老师 001

满足3NF的SCT关系表

巴斯-科德范式 BCNF（续）

- **巴斯-科德范式**（BCNF）：关系R在满足第一范式的基础上，不存在**任何属性**对候选键的传递函数依赖。

【例3.8-续】 $Tno \rightarrow Cno$，$(Sno, Tno) \rightarrow Cno$，$(Sno, Cno) \rightarrow Tno$，所以不满足BCNF。

将SCT分解成两个关系SC和ST以消除主属性对键的部分函数依赖，使之满足BCNF范式。

Sno（学号）	Cno（选课号）	Tno（教师号）
2021310721	CS204	李老师 001
2021310722	MA192	马老师 001
2021310723	CS204	周老师 001

满足3NF的SCT关系表

Sno（学号）	Cno（选课号）
2021310721	CS204
2021310722	MA192
2021310723	CS204

满足BCNF的SC关系表

Tno（教师号）	Cno（选课号）
李老师 001	CS204
马老师 001	MA192
周老师 001	CS204

满足BCNF的TC关系表

巴斯-科德范式 BCNF（续）

- BCNF与3NF的联系比较紧密，主要体现在以下两个特性上：
  （1）如果关系模式*R*满足BCNF，那么该关系模式*R*必定满足3NF；
  （2）如果关系模式*R*满足3NF，且只有一个候选键，那么*R*必定满足BCNF。
- 如果一个关系模式R满足BCNF，则说明在函数依赖的范畴内，它已经实现了关系模式的彻底分解，达到了最高的规范化程度，消除了插入异常和删除异常

巴斯-科德范式 BCNF（续）

- BCNF的关系模式还具备以下性质：
  - 所有非主属性都**完全函数依赖**于每个候选键
  - 所有主属性都**完全函数依赖**于每个不包含它的候选键
  - 没有任何属性**完全函数依赖于非候选键**的任何一组属性

第四范式 4NF

- **第四范式**（4NF）是指关系R在满足巴斯-科德范式的基础上，消除了多值依赖。

- 4NF就是限制关系模式的属性之间不允许有非平凡且非函数依赖的多值依赖。
- 4NF所允许的非平凡多值依赖实际上是函数依赖。

第四范式 4NF（续）

【例3.9】学生联系信息关系Contact，满足3NF，但存在如下的多值依赖：Sno→→Phone，Sno→→Email，不满足4NF，需要对其进行分解。如图3-13（b）（c）所示，可将Contact关系分解成满足4NF的PhoneInfo和EmailInfo关系。

(a) 满足3NF的 Contact 关系表

Sno（学号）	Phone（手机号）	Email（邮箱）
2021310721	137-XXXX-2271	boli@tsinghua.edu.cn
2021310721	137-XXXX-2271	libo@example.com
2021310721	158-XXXX-1790	boli@tsinghua.edu.cn
2021310721	158-XXXX-1790	libo@example.com
2021310722	199-XXXX-2290	yuzhao@tsinghua.edu.cn
2021310722	199-XXXX-2290	zhaoyu1999@exmaple.com

(b) 满足4NF的 PhoneInfo 关系表

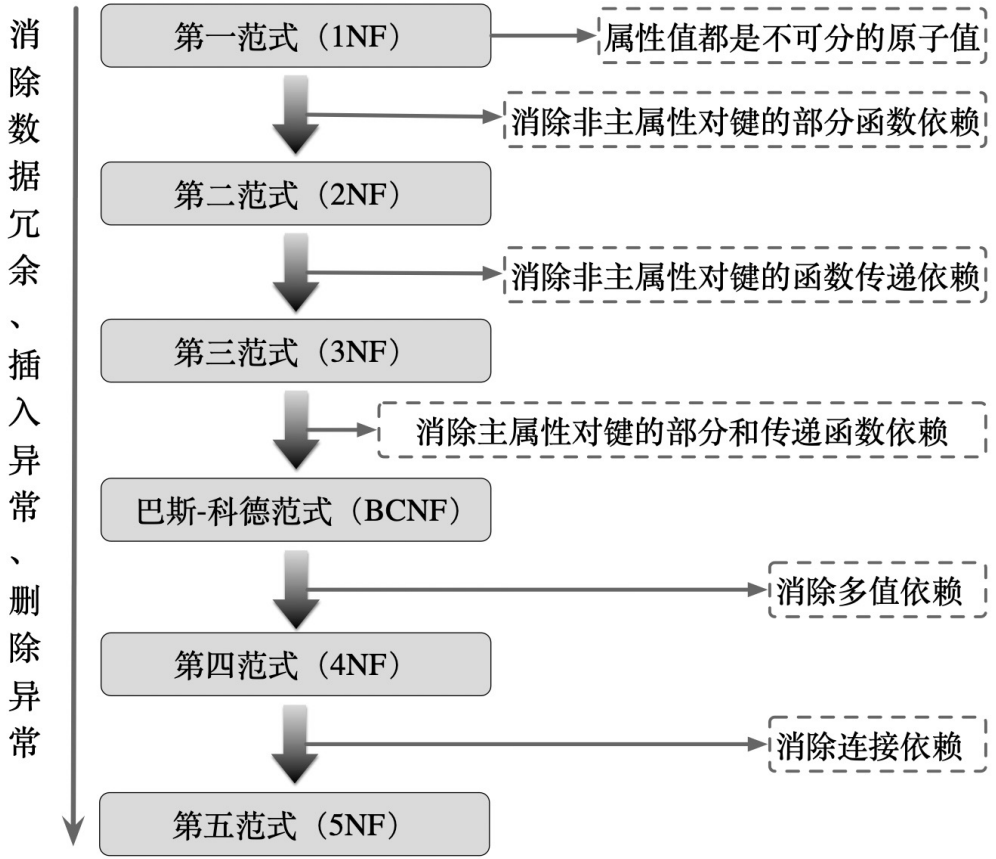
Sno（学号）	Phone（手机号）
2021310721	137-XXXX-2271
2021310721	158-XXXX-1790
2021310722	199-XXXX-2290

(c) 满足4NF的 EmailInfo 关系表

Sno（学号）	Email（邮箱）
2021310721	boli@tsinghua.edu.cn
2021310721	libo@example.com
2021310722	yuzhao@tsinghua.edu.cn
2021310722	zhaoyu1999@exmaple.com

规范化和范式小结

- 越高的范式数据库冗余程度越小
- 高级范式总是包含了低级范式的全部要求
- 最常用的是3NF和BCNF，在实际应用中通常分解到3NF



数据库规范化设计

4、数据库规范化设计理论

4.1 函数依赖理论

4.2 规范化设计和范式

4.3 数据依赖的公理系统

数据依赖的公理系统

- **Armstrong 公理系统**
  - 用于推导关系数据库中的函数依赖和其他数据依赖
  - 提供了一种形式化的方法来理解和处理数据依赖关系，从而提高数据库的可靠性、一致性和性能。
  - 一套推理规则，是模式分解算法的理论基础
  - 可以用于求给定关系模式的键
  - 从一组函数依赖求得蕴涵的函数依赖
- **闭包及其计算**
- **函数依赖集的等价和最小函数依赖集**

Armstrong 公理系统

【定义3.1】**逻辑蕴含**：对于满足一组函数依赖F的关系模式$R<U,F>$，其中任何一个关系r，如果函数依赖$X \rightarrow Y$都成立（即关系r中的任意两元组t, s，若$t[X]=s[X]$，则$t[Y]=s[Y]$），则称F逻辑蕴涵$X \rightarrow Y$，或者称$X \rightarrow Y$是F的逻辑蕴涵。

基于逻辑蕴涵的定义，需要解决的问题：

- 如何从一组函数依赖中推理出蕴涵的函数依赖？
- 如何求得给定关系模式的键？
- **这就需要一组推理规则，这组推理规则就是Armstrong公理系统。**

Armstrong 公理系统（续）

【定义3.2】**Armstrong 公理系统**：$U=\{A_1,A_2,\dots,A_n\}$是R中所有属性的集合，F是U上的一组函数依赖，有关系模式$R<U,F>$，对于$R<U,F>$有以下推理规则：

1. 自反律：若$Y\subseteq X\subseteq U$，则$X\to Y$为F所蕴涵。
2. 增广律：若$X\to Y$为F所蕴涵，且$Z\subseteq U$，则$XZ\to YZ$为F所蕴涵。
3. 传递律：若$X\to Y$和$Y\to Z$为F所蕴涵，则$X\to Z$为F所蕴涵。

根据自反律、增广律和传递律，可以推出以下3条推理规则：

4. 合并规则：若$X\to Y$，$Y\to Z$，则$X\to YZ$。
5. 分解规则：若$X\to Y$，$Z\subseteq \mathrm{Y}$，则$X\to Z$。
6. 伪传递规则：若$X\to Y$，$WY\to Z$，则$XW\to Z$。

Armstrong 公理系统（续）

【引理3.1】基于Armstrong 公理的引理：

$X \to A_1, A_2, \ldots, A_n$成立的充分必要条件是$X \to A_i\ (i = 1,2,3, \ldots, n)$成立。

【例】对于关系模式$R < U, F >$，$U_1, U_2, U_3, U_4, U_5, U_6$是它的属性集$U$的子集，$R$满足函数依赖为$\{U_1 \to U_2U_3, U_3U_4 \to U_5U_6\}$，证明函数依赖$U_1U_4 \to U_6$成立。

证明：

1. 由题中给出$U_1 \to U_2U_3$成立，可以推出$U_1 \to U_3$成立；
2. 根据增广律，可以推出$U_1U_4 \to U_3U_4$成立；
3. 由题中给出$U_3U_4 \to U_5U_6$成立，根据传递律，可以推出$U_1U_4 \to U_5U_6$成立；
4. 可以推出$U_1U_4 \to U_6$成立，题目得证。

函数依赖的公理系统

4.3 数据依赖的公理系统

4.3.1 Armstrong 公理系统

4.3.2 闭包及其计算

4.3.3 函数依赖集的等价和最小函数依赖集

闭包及其计算

在检验范式时，仅考虑给定的函数依赖集是不充分的，还需要考虑在给定的模式上成立的所有函数依赖关系。 基于此，介绍函数依赖集的闭包的定义。

【定义3.3】函数依赖集的闭包：在关系模式$R<U,F>$中，为F所逻辑蕴涵的**函数依赖**的全体叫作F的闭包（closure），记作F^+。

– 简而言之，F^+包含了F中所有的函数依赖。

基于函数依赖集的闭包和Armstrong公理系统，进一步给出属性集闭包的定义：

【定义3.4】属性依赖集的闭包：$U=\{A_1,A_2,\dots,A_n\}$是关系模式R中所有属性的集合，F是U上的一组函数依赖（即R的函数依赖集），$X\subseteq U$，$Y\subseteq U$，$X_F^+=\{A|X\to A$可由F根据Armstrong公理系统推出$\}$，则X_F^+称为**属性集**X关于函数依赖集F的闭包。

闭包及其计算（续）

通过Armstrong公理的引理，可以推导出闭包的引理

【引理3.2】 $U=\{A_1, A_2, \cdots, A_n\}$是关系模式$R$中所有属性的集合，$F$是$U$上的一组函数依赖，$X \subseteq U$，$Y \subseteq U$，$X \rightarrow Y$能由$F$根据Armstrong公理系统推出的充分必要条件是$Y \subseteq X_F^+$。

闭包及其计算（续）

通过上述讨论，可以将判断$X \to Y$能否由F根据Armstrong公理系统推导出来的问题转化为求解X_F^+，判定Y是否为X_F^+的子集问题。

【算法3.1】求解属性集$X(X \subseteq U)$关于U上的函数依赖集F的闭包X_F^+

输入：X，F

输出：X_F^+

算法步骤：计算属性集序列$X^{(i)}(i = 0, 1, \dots)$

步骤（1）令$X^{(0)} = X$，$i = 0$。

步骤（2）求B，$B = \{A | (\exists V)(\exists W)(V \to W \in F \wedge V \subseteq X^{(i)} \wedge A \in W)\}$。

首先，在F中找到尚未用过的左边是$X^{(i)}$的子集的函数依赖：$Y_j \to Z_j(j = 0,1, \dots, k)$，其中$Y_j \subseteq X^{(i)}$。

然后，在Z_j中找到$X^{(i)}$中未出现过的属性构成的属性集B。

步骤（3）令$X^{(i+1)} = B \cup X^{(i)}$。

步骤（4）如果$X^{(i+1)}$和$X^{(i)}$不相等，则$i = i + 1$，返回步骤（2）。

步骤（5）如果$X^{(i+1)}$和$X^{(i)}$相等，或者 $X^{(i)}$和U相等，则$X^{(i)}$就是X_F^+，算法终止。

闭包及其计算（续）

【例】对于关系模式$R<U,F>$，$U=\{A_1,A_2,A_3,A_4,A_5\ A_6\}$是关系模式R中所有属性的集合。$R$的函数依赖集$F=\{A_1\to A_4,A_1A_2\to A_5,A_2A_6\to A_5,A_3A_4\to A_6,A_5\to A_3\}$，$X=A_1A_5$，计算$X_F^+$。

【解】按照算法3.1，依次计算：

（1）令$X^{(0)}=A_1A_5$

（2）在F中找到左边是A_1A_5子集的函数依赖，有$A_1\to A_4$，$A_5\to A_3$，

则$X^{(1)}=X^{(0)}\cup A_4A_3=A_1A_3A_4A_5$，显然$X^{(1)}\neq X^{(0)}$。

（3）在F中找到左边是$A_1A_3A_4A_5$子集的函数依赖，有$A_3A_4\to A_6$，

则$X^{(2)}=X^{(1)}\cup A_6=A_1A_3A_4A_5A_6$。

（4）虽然$X^{(2)}\neq X^{(1)}$，但是F中未用过的函数依赖的左边属性集中已经没有$X^{(2)}$的自己了，因此可以退出循环(没有再计算下去的必要）。

（5）输出$X_F^+=A_1A_3A_4A_5A_6$。

求解关系的候选键

- 在函数依赖部分直观地阐述了如何确定关系的候选键和主键，现在介绍如何基于闭包的计算来求解关系的候选键。
- 对于关系模式$R<U,F>$，其中$U=\{A_1,A_2,\ldots,A_n\}$是R中所有属性的集合，F是U上的一组函数依赖，根据U中属性在F中出现的特点，分成如下四类：
  - L类属性：只在F中某个函数依赖的左部出现。
  - R类属性：只在F中某个函数依赖的右部出现。
  - LR类属性：在F中某个函数依赖的左部和右部均出现。
  - N类属性：在F中每个函数依赖左部和右部均不出现。
- L类和N类属性必然是候选键中的属性
- R类属性必然不是候选键中的属性
- LR类属性有可能是候选键中的属性

求解关系的候选键（续）

求解关系候选键的核心思路

① 求解L和N类属性。

② 若$(L,N)^+ = U$，则(L,N)为唯一候选键。

③ 若$(L,N)^+ \neq U$，则找LR类。

④ 若$(L,N,LR)^+ = U$，则(L,N,LR)为候选键。

求解关系的候选键（续）

【算法3.2】：计算关系的候选键

输入：关系模式$R<U,F>$，其中$U=\{A_1,A_2,\ldots,A_n\}$是R中所有属性的集合。

输出：R的候选键。

算法步骤：

- 步骤（1）根据A_i的特点，划分属性类别（重点检验L和LR类属性）。令X为L类和N类属性集的并集，Y为LR类属性集的集合。
- 步骤（2）根据算法3.1求解属性集X关于U上的函数依赖集F的闭包X_F^+。如果X_F^+包含了R的全部属性，则X是R的唯一候选键，算法结束。否则，转算法步骤（3）
- 步骤（3）遍历Y中的单一属性A，并与X构成属性组XA，如果$(XA)_F^+=U$，则XA为候选键，令$Y=Y-\{A\}$，转算法步骤（4）。
- 步骤（4）如果在上一步找到所有的候选键，则算法结束。否则，遍历Y中的任意两个、三个属性（记为Z），并与X构成属性组XZ，如果$(XZ)_F^+=U$，且XZ不包含已有的候选键，则XZ为新计算的候选键。

求解关系的候选键（续）

【例】设关系模式$R<U,F>$，$U=\{A_1,A_2,A_3,A_4,A_5,A_6\}$，$F=\{A_1A_2\rightarrow A_5, A_1A_3\rightarrow A_6, A_1A_4\rightarrow A_2, A_2\rightarrow A_3, A_3\rightarrow A_4\}$，计算$R$的所有候选键。

解：

- 步骤（1）根据A_i的特点，计算L类、N类、LR类属性。L类属性有A_1，N类属性为空，LR类属性有$A_2A_3A_4$。令X为L类属性的集合，Y为LR类属性的集合。
- 步骤（2）计算属性集X关于U上的函数依赖集F的闭包X_F^+，有$X_F^+=A_1\neq U$。
- 步骤（3）遍历Y中的单一属性，并与X构成属性组，后计算闭包。有：
  - $(XA_2)_F^+=(A_1A_2)_F^+=A_1\ A_2\ A_3\ A_4\ A_5\ A_6$，因此$A_1A_2$是候选键。
  - $(XA_3)_F^+=(A_1A_3)_F^+=A_1\ A_2\ A_3\ A_4\ A_5\ A_6$，因此$A_1A_3$是候选键。
  - $(XA_4)_F^+=(A_1A_4)_F^+=A_1\ A_2\ A_3\ A_4\ A_5\ A_6$，因此$A_1A_4$是候选键。

综上，关系模式$R<U,F>$的所有候选键为A_1A_2、A_1A_3、A_1A_4。

函数依赖的公理系统

4.3 数据依赖的公理系统

4.3.1 Armstrong 公理系统

4.3.2 闭包及其计算

4.3.3 函数依赖集的等价和最小函数依赖集

函数依赖集的等价

- 基于蕴涵的概念，讨论两个函数依赖集的等价和最小函数依赖集的概念
- 如何判断两个函数依赖集是否等价?

【定义3.5】若$F^+ = G^+$，则有函数依赖集F覆盖G（或者G覆盖F），也称F和G等价。

【引理3.3】$F^+ = G^+$的充分必要条件是$F \subseteq G^+$和$G \subseteq F^+$。

【引理证明】根据引理3.3，显然满足必然性。

下面给出充分性的证明：若$F \subseteq G^+$，则$X_F^+ \subseteq X_G^+$，对任意$X \to Y \in F^+$，

有$F^+ \subseteq X_F^+ \subseteq X_G^+$，因此$X \to Y \subseteq (G^+)^+ \subseteq G^+$，即$F^+ \subseteq G^+$。

同理可得，$G^+ \subseteq F^+$，因此有$F^+ = G^+$。

函数依赖集的等价（续）

【例】设F和G是两个函数依赖集，

$F = \{U_1 \to U_2, U_2 \to U_3\}$，$G = \{U_1 \to U_2U_3, U_2 \to U_3\}$，判断它们是否等价。

【解】

- 首先，判断F中的每个函数依赖是否属于G^+。
- 因为$U_{1_G}^+ = U_1U_2U_3$，$U_2 \subseteq U_{1_G}^+$，所以$U_1 \to U_2 \in U_{1_G}^+$。
- 因为$U_{2_G}^+ = U_2U_3$，$U_3 \subseteq U_{2_G}^+$，所以$U_2 \to U_3 \in U_{2_G}^+$。
- 综上，$F \subseteq G^+$。同理可得，$G \subseteq F^+$。
- 根据引理3.3，可得F和G是等价的。

最小函数依赖集（续）

【定义3.6】若函数依赖集F满足以下3个条件，则称F为一个极小函数依赖集，亦称为最小函数依赖集或最小覆盖（minimal cover）。

- F中任一函数依赖的右部仅含一个属性，即右侧是单个属性；
- F中不存在这样一个函数依赖$X \to A$，使得F与$F - \{X \to A\}$等价，即无多余的函数依赖；
- F中不存在这样一个函数依赖$X \to A$，X有真子集Z，使得$F - \{X \to A\} \cup \{Z \to A\}$与$F$等价，即左部无多余的属性。

最小函数依赖集（续）

【例】考虑$R<U,F>$，其中$U=\{Sno,Sdept,Mname,Cno,Grade\}$，

$F=\{Sno\rightarrow Sdept,Sdept\rightarrow Mname,(Sno,\ Cno)\rightarrow Grade\}$。

设$F'=\{Sno\rightarrow Sdept,Sno\rightarrow Mname,Sdept\rightarrow Mname,(Sno,Cno)\rightarrow Grade,(Sno,Sdept)\rightarrow Sdept$。

可以很容易推出F是最小覆盖，而F'不是。因为$F'-\{Sno\rightarrow Mname\}$与$F'$等价，且$F'-\{(Sno,Sdept)\rightarrow Sdept\}$也与$F'$等价。

最小函数依赖集（续）

【定理3.1】每一个函数依赖集F均等价于一个极小函数依赖集F_m。

使用构造性证明的方式，找出F的最小依赖集：

（1）首先依次检查F中的各函数依赖FD_i：使F中每一个函数依赖的右部属性单一化。$X \to Y$，若$Y = A_1A_2A_3, \dots, A_n$(n≥ 2)，则用$\{X \to A_j | (j = 1,2, \dots, n)\}$来取代$X \to Y$。

（2）然后依次检查F中的各函数依赖FD_i：$X \to A$，设$X = B_1B_2B_3, \dots, B_m$(m≥ 2)，逐一检查$B_i(i = 1,2,3, \dots, m)$，若$B_i \in (X - Bi)_F^+$，则用$X - B_i$代替$X$。

（3）最后依次检查F中的各函数依赖FD_i：$X \to A$，令$G = F - \{X \to A\}$，若$A \in X_G^+$，则从F中去掉此依赖。

- **F的最小函数依赖集不一定是唯一的，它与对各函数依赖FD_i以及$X \to A$中X各属性的处理顺序有关。**
- **定理的证明过程（极小化过程）也是检验F是否为极小依赖集的一个算法。**

最小函数依赖集（续）

【例】对于关系模式$R<U,F>$的函数依赖集：

$$F=\{A\rightarrow B, B\rightarrow A, B\rightarrow C, A\rightarrow C, C\rightarrow A\}$$，F的最小依赖集有：

- $F_{m1}=\{A\rightarrow B, B\rightarrow C, C\rightarrow A\}$
- $F_{m2}=\{A\rightarrow B, B\rightarrow A,\ A\rightarrow C,\ C\rightarrow A\}$。

注意，F的最小函数依赖不一定是唯一的。

目录

数据库规范化设计实现

- 关系模式的规范化过程是通过对关系模式的分解来实现的。
- 将满足低一级范式的关系模式分解成若干个满足高一级范式的关系模式的方法并不是唯一的，因此需要规范化算法。
- 数据库规范化实现是基于数据依赖公理系统，通过关系模式的分解算法，这些分解算法可以保证分解后的关系模式与原关系模式等价

关系模式的分解

5、数据库规范化设计实现

5.1 关系模式分解的定义

5.2 分解的无损连接性

5.3 分解的保持依赖性

5.4 模式分解算法

关系模式分解的定义

给定：$\mathrm{R}=\{A,B,C\}, F=\{A\rightarrow B, B\rightarrow C\}$

分解为：$\mathrm{R}_1=\{A,B\}$，$\mathrm{R}_2=\{B,C\}$

【定义3.7】对于关系模式$R<U,F>$，它的一个分解是指

$\rho=\{R_1<U_1,F_1>,R_2<U_2,F_2>,\ldots,R_n<U_n,F_n>\}$，并且满足

① 关系不丢失：$U=\bigcup_{i=1}^{n}U_i$

② 模式不冗余：不存在$U_i\subseteq U_j(1\leq i,j\leq n)$

③ 依赖不丢失：F_i是F在U_i上的投影，即$F_i=\{X\rightarrow Y|X\rightarrow Y\in F^+\wedge XY\subseteq U_i\}$。

关系模式分解的定义（续）

- 对于一个关系模式$R<U,F>$的分解方式有多种，但都遵循一个原则：分解之后的若干模式$\rho=\{R_1<U_1,F_1>,R_2<U_2,F_2>,\ldots,R_n<U_n,F_n>\}$与原模式$R<U,F>$等价。

- 关系模式等价分解的概念可以从以下两个角度进行考虑：

（1）数据等价：分解具有无损连接性（lossless join），无损连接是指分解后的关系通过自然连接可以恢复分解前的关系，即通过自然连接得到的关系与分解前的关系相比，既不多出信息、又不丢失信息；

（2）语义等价：分解要保持函数依赖（preserve functional dependency）。

因此关系模式等价分解既要保持函数依赖，又要具有无损连接性。

关系模式的分解

5、数据库规范化设计实现

5.1 关系模式分解的定义

5.2 分解的无损连接性

5.3 分解的保持依赖性

5.4 模式分解算法

分解的无损连接性

【定义3.8】具有无损的模式分解：关系模式$R<U,F>$的一个分解是$\rho=\{R_1<U_1,F_1>,R_2<U_2,F_2>,\ldots,R_n<U_n,F_n>\}$，若$R_1$、$R_2$、…、$R_n$自然连接的结果相等，则这个分解$\rho$具有无损连接性，$\rho$是关系模式$R<U,F>$无损分解。

- 虽然无损分解保证不丢失信息，但是无损分解不一定能解决插入异常、删除异常、修改复杂和数据冗余等问题。

分解的无损连接性（续）

【例】如下图所示，对于关系模式$R<\{\mathbf{Sno},\mathbf{Sdept},\boldsymbol{Sloc}\},\{\mathbf{Sno}\rightarrow\mathbf{Sdept},\mathbf{Sno}\rightarrow\boldsymbol{Sloc}\}>$，有两种分解方式：

$\rho_1=\{R_1=\prod_{Sno\ Sdept}(R),\ R_2=\prod_{Sno\ Sloc}(R)\}$，

$\rho_2=\{R_1=\prod_{Sno\ Sdept}(R),\ R_3=\prod_{Sdept\ Sloc}(R)\}$。

这两种分解方式的自然连接结果如图2-17所示，不难发现：

分解方式$\rho_1=\{\prod_{Sno\ Sdept}(R),\ \prod_{Sno\ Sloc}(R)\}$是无损分解

分解方式$\rho_2=\{\prod_{Sno\ Sdept}(R),\ \prod_{Sdept\ Sloc}(R)\}$不是无损分解

R

Sno（学号）	Sdept（所在系）	Sloc(宿舍)
2021310721	CS	CS_A107
2021310722	MA	MA_B22
2021310723	CS	CS_B23

(a) 关系R

$R_1=\prod_{Sno\ Sdept}(R)$

Sno（学号）	Sdept（所在系）
2021310721	CS
2021310722	MA
2021310723	CS

$R_2=\prod_{Sno\ Sloc}(R)$

Sno（学号）	Sloc(宿舍)
2021310721	CS_A107
2021310722	MA_B22
2021310723	CS_B23

(b) 分解方式1

$R_1=\prod_{Sno\ Sdept}(R)$

Sno（学号）	Sdept（所在系）
2021310721	CS
2021310722	MA
2021310723	CS

$R_3=\prod_{Sdept\ Sloc}(R)$

Sdept（所在系）	Sloc(宿舍)
CS	CS_A107
MA	MA_B22
CS	CS_B23

(c) 分解方式2

分解的无损连接性（续）

【例-续】这两种分解方式的自然连接结果如下右图所示，

分解方式$\rho_1 = \{\prod_{Sno\ Sdept}(R), \prod_{Sno\ Sloc}(R)\}$是无损分解

分解方式$\rho_2 = \{\prod_{Sno\ Sdept}(R), \prod_{Sdept\ Sloc}(R)\}$不是无损分解。

R

Sno（学号）	Sdept（所在系）	Sloc(宿舍)
2021310721	CS	CS_A107
2021310722	MA	MA_B22
2021310723	CS	CS_B23

(a) 关系R

$R_1 = \prod_{Sno\ Sdept}(R)$

Sno（学号）	Sdept（所在系）
2021310721	CS
2021310722	MA
2021310723	CS

$R_2 = \prod_{Sno\ Sloc}(R)$

Sno（学号）	Sloc(宿舍)
2021310721	CS_A107
2021310722	MA_B22
2021310723	CS_B23

(b) 分解方式1

$R_1 = \prod_{Sno\ Sdept}(R)$

Sno（学号）	Sdept（所在系）
2021310721	CS
2021310722	MA
2021310723	CS

$R_3 = \prod_{Sdept\ Sloc}(R)$

Sdept（所在系）	Sloc(宿舍)
CS	CS_A107
MA	MA_B22
CS	CS_B23

(c) 分解方式2

关系R的两种分解方式

$\prod_{Sno\ Sdept}(R) \bowtie \prod_{Sno\ Sloc}(R)$

Sno（学号）	Sdept（所在系）	Sloc(宿舍)
2021310721	CS	CS_A107
2021310722	MA	MA_B22
2021310723	CS	CS_B23

(a) 分解方式1的自然连接结果

$\prod_{Sno\ Sdept}(R) \bowtie \prod_{Sdept\ Sloc}(R)$

Sno（学号）	Sdept（所在系）	Sloc(宿舍)
2021310721	CS	CS_A107
2021310721	CS	CS_B23
2021310722	MA	MA_B22
2021310723	CS	CS_A107
2021310723	CS	CS_B23

(b) 分解方式2的自然连接结果

两种分解方式的自然连接结果

启发：一般不能直接由定义判断一个分解是否为无损分解

检验无损分解的算法

【算法3.3】：检验一个分解是否为无损分解

输入：关系模式$R<\{A_1, A_2, \ldots, A_n\}, F>$, $\rho=\{R_1, R_2, \ldots, R_m\}$

输出：确定ρ是否为无损分解

算法步骤：

- 步骤（1）构造一个n列m行的表，每一列对应于属性，每一行对应于分解中的一个关系模式。
- 步骤（2）如果$A_j \in R_i$，则在第 j 列第 i 行上置为a_i，否则置为b_{ij}。
- 步骤（3）依次检查F中的每一个函数依赖，并修改表中的元素。对于F中的一个函数依赖$X \rightarrow Y$，在X的分量上寻找相同的行，然后将这些行的Y分量赋相同符号，如果其中有a_j，则将b_{ij}改为a_j，反之则改为b_{ij}。
- 步骤（4）$a_1 a_2 \ldots a_n$，则ρ是无损分解，算法结束，退出；否则进行步骤（5）。
- 步骤（5）如果F中所有的函数依赖都不能再修改表，且没有发现某行变成$a_1 a_2 \ldots a_n$，则ρ不是无损分解，算法结束，退出。

检验无损分解的算法（续）

【例】如下图所示，对于关系模式$R<\{\mathbf{Sno},\mathbf{Sdept},\boldsymbol{Sloc}\},\{\mathbf{Sno}\rightarrow\mathbf{Sdept},\mathbf{Sno}\rightarrow\boldsymbol{Sloc}\}>$，有两种分解方式：

$\rho_1=\{R_1=\prod_{Sno\ Sdept}(R),\ R_2=\prod_{Sno\ Sloc}(R)\}$，

$\rho_2=\{R_1=\prod_{Sno\ Sdept}(R),\ R_3=\prod_{Sdept\ Sloc}(R)\}$。

请使用算法3.2检验分解方式ρ_1,ρ_2的无损连接性，即判断分解方式ρ_1,ρ_2是否为无损连接。

R

Sno（学号）	Sdept（所在系）	Sloc(宿舍)
2021310721	CS	CS_A107
2021310722	MA	MA_B22
2021310723	CS	CS_B23

(a) 关系R

关系R的两种分解方式

$R_1=\prod_{Sno\ Sdept}(R)$

Sno（学号）	Sdept（所在系）
2021310721	CS
2021310722	MA
2021310723	CS

$R_2=\prod_{Sno\ Sloc}(R)$

Sno（学号）	Sloc(宿舍)
2021310721	CS_A107
2021310722	MA_B22
2021310723	CS_B23

(b) 分解方式1

$R_1=\prod_{Sno\ Sdept}(R)$

Sno（学号）	Sdept（所在系）
2021310721	CS
2021310722	MA
2021310723	CS

$R_3=\prod_{Sdept\ Sloc}(R)$

Sdept（所在系）	Sloc(宿舍)
CS	CS_A107
MA	MA_B22
CS	CS_B23

(c) 分解方式2

检验无损分解的算法（续）

【解】对于分解方式$\rho_1 = \{R_1 = \prod_{Sno\ Sdept}(R),\ R_2 = \prod_{Sno\ Sloc}(R)\}$，使用算法3.3检验：

算法步骤（1）：构造初始表，如下图（a）所示。

算法步骤（2）：进行初始化赋值。

- 对于R_1，包括两个属性Sno，Sdept，因此第1行Sno，Sdept列的值分别为a_1, a_2，因为$Sloc$不在R_1中，因此该列对应的单元格值为$b_{1,3}$。
- 对于R_2，包括两个属性Sno，Sloc，因此第2行Sno，Sloc列的值分别为a_1, a_3，因为Sdept不在R_2中，因此该列对应的单元格值为$b_{2,2}$。

R_i	Sno（学号）	Sdept（所在系）	Sloc（宿舍）
$\prod_{Sno\ Sdept}(R)$	a_1	a_2	$b_{1,3}$
$\prod_{Sno\ Sloc}(R)$	a_1	$b_{2,2}$	a_3

(a) 构造初始表

R_i	Sno（学号）	Sdept（所在系）	Sloc（宿舍）
$\prod_{Sno\ Sdept}(R)$	a_1	a_2	$b_{1,3}$
$\prod_{Sno\ Sloc}(R)$	a_1	$\boldsymbol{a_2}$	a_3

(b) 检查$Sno \rightarrow Sdept$并更新表元素

检验ρ_1的无损连接性

检验无损分解的算法（续）

【解-续】

算法步骤（3）：依次检查F中的每一个函数依赖，并修改表中的元素。

检查Sno → Sdept，由于R_1、R_2的Sno列相同，因此将R_2的Sdept列修改为a_2（下图(b)）

算法步骤（4）：此时发现第2行都变成a_1, a_2, a_3，**可以判断ρ_1是无损分解，算法结束。**

R_i	Sno（学号）	Sdept（所在系）	Sloc(宿舍)
$\Pi_{Sno,Sdept}(R)$	a_1	a_2	$b_{1,3}$
$\Pi_{Sno,Sloc}(R)$	a_1	$b_{2,2}$	a_3

(a) 构造初始表

R_i	Sno（学号）	Sdept（所在系）	Sloc(宿舍)
$\Pi_{Sno,Sdept}(R)$	a_1	a_2	$b_{1,3}$
$\Pi_{Sno,Sloc}(R)$	a_1	$\boldsymbol{a_2}$	a_3

(b) 检查$Sno \rightarrow Sdept$并更新表元素

检验ρ_1的无损连接性

检验无损分解的算法（续）

【解-续】对于分解方式$\rho_2 = \{R_1 = \prod_{Sno\ Sdept}(R),\ R_3 = \prod_{Sdept\ Sloc}(R)\}$，使用算法3.2检验：

算法步骤（1）：构造初始表，如下图（a）所示。

算法步骤（2）：进行初始化赋值。

- 对于R_1，包括两个属性Sno，Sdept，因此第1行Sno，Sdept列的值分别为a_1, a_2，因为$Sloc$不在R_1中，因此该列对应的单元格值为$b_{1,3}$。
- 对于R_2，包括两个属性Sdept，Sloc，因此第2行Sdept，Sloc列的值分别为a_2, a_3，因为Sno不在R_2中，因此该列对应的单元格值为$b_{2,1}$。

R_i	Sno（学号）	Sdept（所在系）	Sloc(宿舍)
$\prod_{Sno\ Sdept}(R)$	a_1	a_2	$b_{1,3}$
$\prod_{Sdept\ Sloc}(R)$	$b_{2,1}$	a_2	a_3

(a) 构造初始表

R_i	Sno（学号）	Sdept（所在系）	Sloc(宿舍)
$\prod_{Sno\ Sdept}(R)$	a_1	a_2	$b_{1,3}$
$\prod_{Sdept\ Sloc}(R)$	$b_{2,1}$	a_2	a_3

(b) 检查$Sno \rightarrow Sdept$并更新表元素

R_i	Sno（学号）	Sdept（所在系）	Sloc(宿舍)
$\prod_{Sno\ Sdept}(R)$	a_1	a_2	$b_{1,3}$
$\prod_{Sdept\ Sloc}(R)$	$b_{2,1}$	a_2	a_3

(c) 检查$Sno \rightarrow Sloc$并更新表元素

检验ρ_2的无损连接性

检验无损分解的算法（续）

【解-续】

算法步骤（3）：依次检查F中的每一个函数依赖，并修改表中的元素。

- 检查Sno → Sdept，由于R_1、R_2的Sno列的值不相同，因此不对表中元素修改（下图（b））。
- 检查Sno → Sloc，同理也找不到相同值，因此不对表中元素修改（下图（c））。

算法步骤（4）：没有找到任何值为$a_1a_2a_3$的行。

算法步骤（5）：已经遍历了所有函数依赖，并且都不能再修改表的元素值，且找到任何值为$a_1a_2a_3$的行，**可以判断ρ_2不是无损分解。算法结束。**

R_i	Sno（学号）	Sdept（所在系）	Sloc(宿舍)
$\prod_{Sno\ Sdept}(R)$	a_1	a_2	$b_{1,3}$
$\prod_{Sdept\ Sloc}(R)$	$b_{2,1}$	a_2	a_3

(a) 构造初始表

R_i	Sno（学号）	Sdept（所在系）	Sloc(宿舍)
$\prod_{Sno\ Sdept}(R)$	a_1	a_2	$b_{1,3}$
$\prod_{Sdept\ Sloc}(R)$	$b_{2,1}$	a_2	a_3

(b) 检查$Sno \rightarrow Sdept$ 并更新表元素

R_i	Sno（学号）	Sdept（所在系）	Sloc(宿舍)
$\prod_{Sno\ Sdept}(R)$	a_1	a_2	$b_{1,3}$
$\prod_{Sdept\ Sloc}(R)$	$b_{2,1}$	a_2	a_3

(c) 检查 $Sno \rightarrow Sloc$ 并更新表元素

检验ρ_2的无损连接性

保持分解无损依赖的定理

- 定理：设$\rho=\{R_1, R_2\}$是关系模式R的一个分解，F是R的函数依赖集，那么ρ是R（关于F）的无损分解的充分必要条件是：

  $(R_1 \cap R_2) \rightarrow R_1 - R_2 \in F^+$ 或 $(R_1 \cap R_2) \rightarrow R_2 - R_1 \in F^+$

- 定理：设F是关系模式R的函数依赖集，$\rho=\{R_1, R_2, \ldots, R_k\}$是$R$关于$F$的一个无损联接分解。若$\sigma=\{S_1, S_2, \ldots, S_m\}$是$R_i$关于$F_i=\pi_{Ri}(F)$一个无损联接分解，则$\rho' =\{R_1, \ldots, R_{i-1}, S_1, S_2, \ldots, S_m, R_{i+1}, \ldots, R_k\}$是$R$关于F的无损联接分解。

- 定理：设F是关系模式R的函数依赖集，$\rho=\{R_1, R_2, \ldots, R_k\}$是$R$关于$F$的一个无损联接分解。设$\rho^+=\{R_1, \ldots, R_k, R_{k+1}\} \supseteq \rho$是R的一个分解，$\rho^+$也是$R$关于$F$的无损联接分解。

关系模式的分解

5、 数据库规范化设计实现

5.1 关系模式分解的定义

5.2 分解的无损连接性

5.3 分解的保持依赖性

5.4 模式分解算法

分解的函数依赖保持

【定义3.9】保持函数依赖的模式分解：如果$F^+ = (\cup_{i=1}^{n} F_i)^+$，则关系模式$R < U, F >$的一个分解$\rho = \{R_1 < U_1, F_1 >, R_2 < U_2, F_2 >, \ldots, R_n < U_n, F_n >\}$保持函数依赖。

- 满足的性质：
  - 若一个分解具有无损连接性，则它能够保证不丢失信息；
  - 若一个分解保持了函数依赖，则它可以减轻或者解决各种异常情况；
  - 然而，无损分解和保持函数依赖的分解是两个相互独立的标准。
    - 具有无损连接性的分解不一定能够保持函数依赖；
    - 保持函数依赖的分解也不一定是无损分解。

分解的保持依赖性（续）

【例】有关系模式$R<U=\{U_1,U_2,U_3,U_4,U_5\},F=\{U_1\to U_3,U_2\to U_3,U_3\to U_4,U_4U_5\to U_3,U_3U_5\to U_1\}>$，有分解方式：$\rho_1=\{R_1=\prod_{U_1,U_3}(R),\ R_2=\prod_{U_2,U_3}(R),R_3=\prod_{U_3,U_4,U_5}(R)\}$，判断$\rho_1$是否保持函数依赖。

【解】

- 1. 求关系模式R的函数依赖集F在各分解关系中的投影。
  - $\prod_{R_1}(F)=\{U_1\to U_3\}$,
  - $\prod_{R_2}(F)=\{U_2\to U_3\}$,
  - $\prod_{R_3}(F)=\{U_3\to U_4,U_4U_5\to U_3\}$
- 2. 求所有投影的并集G：$G=\{U_1\to U_3,U_2\to U_3,U_3\to U_4,U_4U_5\to U_3\}\neq F$
- 3. 因此ρ_1不保持函数依赖。

关系模式的分解

5、数据库规范化设计实现

5.1 关系模式分解的定义

5.2 分解的无损连接性

5.3 分解的保持依赖性

5.4 模式分解算法

模式分解算法

- **首先回顾之前提到的一个好的关系数据库的三个设计目标：**

  （1）满足BCNF范式（应该至少满足3NF ）；

  （2）具有无损连接性；

  （3）保持函数依赖。

- **为了达到上述的三个目标，通常需要对不满足这些目标的关系模式进行分解。**
- 将满足低一级范式的关系模式分解成若干个满足高一级范式的关系模式的方法并不是唯一的。
- 为了解决这个问题，基于数据依赖公理系统，介绍若干种关系模式的分解算法。
- 这些分解算法可以保证分解后的关系模式与原关系模式等价。

模式分解算法（续）

➢ **对于模式分解：**

（i）如果要求分解保持函数依赖，模式分解可以达到3NF，但不一定满足BCNF；

（ii）如果要求分解既保持函数依赖，又具有无损连接性，那么模式分解可以达到3NF，但不一定满足BCNF；

（iii）如果要求分解具有无损连接性，则模式分解一定可以达到4NF；

- 上述（i）和（ii）分别由算法3.4、算法3.5和算法3.6实现。
- 对于（iii），可以基于算法3.6转换为BCNF的无损连接分解后，针对其中不满足4NF的关系模式，可以再进一步分解，使得每一关系模式都满足4NF，此时分解的结果都是满足4NF的无损连接分解。

模式分解算法（续）

【算法3.4】分解到3NF，并保持函数依赖的模式分解算法。

输入： 关系模式$R<U,F>$

输出： $\rho=\{R_1<U_1,F_1>,R_2<U_2,F_2>,\ldots,R_m<U_m,F_m>\}$保持函数依赖

算法步骤：

- 步骤（1）令$\rho=\emptyset$，计算F的最小函数依赖集F_m。
- 步骤（2）令$U_0=\emptyset$，对U中的每个属性A_i，若其不出现在F_m中任一函数依赖的左端和右端，令U_0=$U_0\cup\{A_i\}$。以U_0为属性集，构造一个新的关系模式R_0，令$\rho=\rho\cup\{R_0\}$，$U=U-U_0$。
- 步骤（3）若$X\rightarrow Y\in F_m$，且$XY=U$，则输出$\rho=\rho\cup\{R\}$。即R为3NF，转步骤(6)；否则转步骤(4)。
- 步骤（4）若F_m中存在左端相同的函数依赖$X\rightarrow Y_1,X\rightarrow Y_2,\ldots,X\rightarrow Y_m$；对其进行合并，令$F_m=(F_m-\{X\rightarrow Y_1,X\rightarrow Y_2,\ldots,X\rightarrow Y_m\})\cup\{X\rightarrow(Y_1\cup Y_2\cup\cdots\cup Y_m,)\}$。重复步骤（4），直到$F_m$中不存在左端相同的函数依赖。
- 步骤（5）对F_m中每个函数依赖$X_i\rightarrow Y_i$，令$U_{\mathrm{i}}=X_i\cup Y_i$，构造$R_{\mathrm{i}}(U_{\mathrm{i}})$，令$\rho=\rho\cup\{R_{\mathrm{i}}\}$。
- 步骤（6）算法终止，输出ρ。

模式分解算法（续）

□算法3.4的核心思想是通过逐步拆分和合并关系模式，以逐步合成若干个满足3NF且保持函数依赖的关系模式。通过算法3.4得到的$\{R_1 < U_1, F_1 >, R_2 < U_2, F_2 >, \ldots, R_m < U_m, F_m >\}$一定都满足3NF吗？

【算法3.4正确性的简要证明】

– 设$F_i' = \{X \to Y_1, X \to Y_2, \ldots, X \to Y_m\}, U_i = \{X, Y_1, Y_2, \ldots, Y_m\}$，有：

1. 由F_i'可得$R_i < U_i, F_i >$一定是以X为键。
2. 假设$R_i < U_i, F_i >$不满足3NF，则必存在非主属性$Y_k (1 \le k \le m)$和属性组合U，$Y_k \in U$，使得$X \to U$，$U \to Y_k \in F_i^+$，而$U \to X \notin F_i^+$。

– 假设$U \subset X$，则与$X \to Y_k$属于最小函数依赖集F是矛盾的。因此，$U \not\subseteq X$。

– 设$U \cap X = X_1$，$U - X = \{Y_1, \ldots, Y_p\}$，令$G = F - \{X \to Y_k\}$，可得$U \subseteq X_G^+$，即$X \to U \in G^+$。同理可得$U \to Y_k \in G^+$。因此，$X \to Y_k \in G^+$。若$X \to Y_k \in G^+$成立，则与F是最小函数依赖集的前提相矛盾，因此$R_i < U_i, F_i >$满足3NF。

模式分解算法（续）

【算法3.5】分解到3NF，既保持无损连接性又保持函数依赖的模式分解。

输入： 关系模式$R<U,F>$

输出： $\rho=\{R_1<U_1,F_1>,R_2<U_2,F_2>,\ldots,R_m<U_m,F_m>\}$保持无损连接性又保持函数依赖。

算法步骤：

步骤（1）基于算法3.4，求出保持函数依赖的分解$\rho=\{R_1<U_1,F_1>,R_2<U_2,F_2>,\ldots,R_k<U_k,F_k>\}$。

步骤（2）选择$R<U,F>$的主键X，将主键与函数依赖相关的属性组成一个新的关系模式$\boldsymbol{R_{k+1}<X,F_X>}$。

步骤（3）若有某个U_i，满足$X\subseteq U_i$，则输出ρ；否则，输出$\rho\cup\{\boldsymbol{R_{k+1}}\}$。

模式分解算法（续）

- 通过算法3.5可以看出，该算法首先基于算法3.4分解出若干满足3NF且保持函数依赖的关系模式ρ。
- 因为ρ保持函数依赖是显然的且R_{k+1}也是显然满足3NF的，因此算法只需要检验ρ的无损连接性即可（步骤（3））。
- 由于$\rho \cup \{R_{k+1}\}$中必然存在一个关系模式R(T)的属性组$T \supseteq X$，X是R_{k+1}的键，任意取$U - T$中的属性B，必存在某个i，使得$B \in T^{(i)}$（根据求函数依赖集闭包的算法）。然后对i实施归纳法，可以基于算法3.2检验其无损连接性。

模式分解算法（续）

【例】设有关系模式$R<U,F>, U=\{A,B,C,D,E,F\}$，$F=\{A\to BE, AE\to F, BC\to D, D\to A\}$，试将$R$分解为3NF，且具有依赖保持性及无损连接性。

（1）构造F的最小依赖集F_m。

- 将F右部分解为单属性：$\{A\to B, A\to E, AE\to F, BC\to D, D\to A\}$
- 去掉多余函数依赖：$\{A\to B, A\to E, AE\to F, BC\to D, D\to A\}$
- 去掉左部分多余属性：$\{A\to B, A\to E, A\to F, BC\to D, D\to A\}$

（2）在最小依赖集F_m中找到$X\to A\in F_m$且$XA=U$的函数依赖。（不存在）

（3）找出在最小依赖集F_m中未出现的属性。（不存在）

（4）对最小依赖集F_m中每组左部相同的函数依赖，构造R_i，$\rho=\{ABEF, BCD, AD\}$

（5）求R的键。

- 求在最小依赖集F_m右边尚未出现过的属性集合：$\{C\}$。
- 若$\{C\}$不是R的键，则求小依赖集F_m左右都出现过的属性集合：$\{A,B,D\}$。
- 判断AC, BC, CD是否为R的键。

（6）因为AC, BC, CD是R的键，且$BC\subseteq BCD$，所以$\rho=\{ABEF, BCD, AD\}$即为目标分解。

模式分解算法（续）

【算法3.6】分解到BCNF，具有无损连接性的模式分解算法。

输入： 关系模式$R<U,F>$

输出： $\rho=\{R_1<U_1,F_1>,R_2<U_2,F_2>,\ldots,R_m<U_m,F_m>\}$保持无损连接性。

算法步骤：

步骤（1）令$\rho=\{R<U,F>\}$。

步骤（2）若ρ中各个关系模式都满足BCNF，则算法终止，转步骤（4），否则步骤（3）。

步骤（3）任取ρ中不满足BCNF的关系模式$R_{\mathrm{i}}(U_{\mathrm{i}})$，$F$在$U_{\mathrm{i}}$上的投影为$F_{\mathrm{i}}$，由于$R_{\mathrm{i}}(U_{\mathrm{i}})$不满足BCNF，则必定存在函数依赖$X\rightarrow Y\in F_i^+$，其中$X$不是$R_{\mathrm{i}}(U_{\mathrm{i}})$的候选键，且$Y\not\subseteq X$。然后，分别以属性集$U_{\mathrm{i}}-\{Y\}$和$X\cup Y$构造关系模式$R_i'$和$R_i''$，令$\rho=(\rho-\{R_{\mathrm{i}}\})\cup\{R_i',R_i''\}$，转步骤（2）。

步骤（4）算法终止，输出ρ（由于U中的属性个数有限，该算法经有限次循环后必然终止）。

模式分解算法（续）

【算法3.6】分解到BCNF，具有无损连接性的模式分解算法。

- 通过观察算法3.5，不难发现该算法是一个自顶向下的算法
- 因为它从根节点（初始的关系模式$R<U,F>$）开始（步骤（1））， 当检验发现当前拆分出的关系模式不满足BCNF时（步骤（2）），自然地构建了一棵对初始的关系模式$R<U,F>$的二叉分解树，逐步拆分成更小的关系模式（步骤（3）），直到满足BCNF条件。
- 值得注意的是，$R<U,F>$的分解并不是唯一的，这与步骤（3）中选择具体的$X \rightarrow Y$有关。

模式分解算法（续）

【例】设有关系模式$R<U,F>$，

$U=\{\text{CourseId, CourseName, TeacherId, CourseTime, CourseRoom}\}$，

$F=\{\text{CourseId} \rightarrow \text{TeacherId}, \text{TeacherId} \rightarrow \text{CourseTime}, \text{CourseName} \rightarrow \text{TeacherId},$
$(\text{CourseTime, CourseRoom}) \rightarrow \text{TeacherId}, (\text{TeacherId, CourseRoom}) \rightarrow \text{CourseId}\}$。

请将R分解为具有无损连接性的BCNF。

【解】令$\rho=\{R<U,F>\}$。

- ρ 中不是所有的关系模式都是BCNF，转入下一步。
- 分解R: 因为F中所有函数依赖的右边没有CourseName, CourseRoom，因此，R上的候选键包含CourseName, CourseRoom。考虑$\text{CourseId} \rightarrow \text{TeacherId}$函数依赖不满足BCNF条件，可将其分解成$R_1(\text{CourseId, TeacherId})$和
  $R_2(\text{CourseId, CourseName, CourseTime, CourseRoom})$。
- 计算R_1和R_2的最小函数依赖集：$F_1=\{\text{CourseId} \rightarrow \text{TeacherId}\}, F_2=\{\text{CourseName} \rightarrow \text{CourseTime}, (\text{CourseName, CourseRoom}) \rightarrow \text{CourseId}\}$.

模式分解算法（续）

【例】设有关系模式$R<U,F>$，

$U=\{\text{CourseId, CourseName, TeacherId, CourseTime, CourseRoom}\}$，

$F=\{\text{CourseId} \rightarrow \text{TeacherId}, \text{TeacherId} \rightarrow \text{CourseTime}, \text{CourseName} \rightarrow \text{TeacherId},$
$(\text{CourseTime, CourseRoom}) \rightarrow \text{TeacherId}, (\text{TeacherId, CourseRoom}) \rightarrow \text{CourseId}\}$。

请将R分解为具有无损连接性的BCNF。

【解续】

- 分解R_2：因为R_2 (CourseId, CourseName, CourseTime, CourseRoom)的候选键为(CourseName, CourseRoom)。考虑CourseName → CourseTime不满足BCNF条件，将其进一步分解为R_{21}(CourseName, CourseTime)和R_{22}(CourseId, CourseName, CourseRoom).
- 计算R_{21}和R_{22}的最小函数依赖集分别为$F_{21}=\{\text{CourseName} \rightarrow \text{CourseTime}\}$，$F_{22}=\{(\text{CourseName, CourseRoom}) \rightarrow \text{CouserId}\}$。由于$R_{22}$的候选键为{CourseName, CourseRoom}，F_{22}中的所有函数依赖满足BCNF条件。
- 算法终止，输出：$\rho=\{R_1(\text{CourseId, TeacherId}), R_{21}(\text{CourseName, CourseTime}), R_{22}(\text{CourseId, CourseName, CourseRoom})\}$

目录

本章小结

➢ 数据库设计和数据模型

- 在数据库设计过程中，主要经历了概念结构设计、逻辑结构设计和物理结构设计三个阶段，这三个阶段使用的数据模型分别称为概念数据模型、逻辑数据模型和物理数据模型。
- 在关系数据库的设计中，通常是使用E-R模型进行概念结构设计，使用关系模型进行逻辑结构设计。
- 因此，关系数据库设计的一个核心步骤就是将E-R模型转换为关系模型。

本章小结（续）

E-R模型

- 实体是用户对现实世界中事物数据概念的某种抽象。
- 关系是具有一定属性特征的二维表，是用于抽象表示存储实体的数据结构。
- 在关系数据库中，表、关系、关系表和实体集是同义词。
- 实体集可以被一组数据特征来描述实体集的属性（也是关系的属性）。

E-R模型的基本元素包括实体集、属性和联系

- 联系是指实体集之间的联系
- E-R模型中的联系有不同的类型，常见的有一元联系、二元联系和多元联系。
- 在实际应用中，二元联系是最常见的联系类型。
- 对于二元联系，细分成以下几种类型：一对一联系、一对多联系和多对多联系。

本章小结（续）

➢从E-R图到关系设计

- 简要地说，将E-R模型转换为关系模型即是：
  - ◆将每一个实体集转换成一个关系表
  - ◆实体集的属性转换为关系表的列
  - ◆实体集的标识符转换为关系表的键
  - ◆并将联系转换为一个关系模式，该关系模式的属性就是该联系所关联的实体集的标识符的集合以及该联系自身的属性。

本章小结（续）

➢ 数据库规范化设计理论

- 关系表的规范化设计就是要尽可能地减少关系表中列或者列组之间的依赖关系，进而得到简洁独立的关系表，其实现方法就是避免数据冗余。
- 范式是指关系表设计的规范化程度。
  - ◆不同的范式要求可以设计出冗余程度不同的数据库。
  - ◆目前关系数据库有六类范式：第一范式（1NF）、第二范式（2NF）、第三范式（3NF）、巴斯-科德范式（BCNF）、第四范式（4NF）和第五范式（5NF）。
  - ◆各种范式呈递次规范，越高的范式数据库冗余程度越小，高级范式总是包含了低级范式的全部要求，满足最低要求的范式是第一范式（1NF）。
- 一般来说，数据库只需满足第三范式（3NF）或巴斯-科德范式（BCNF）。

本章小结（续）

- **数据依赖的公理系统**
  - 函数依赖是关系数据库规范设计的理论基础
  - Armstrong 公理系统给出了函数依赖的自反律、增广律和传递律。
  - Armstrong 公理系统是有效且完备的。
- **关系模式的分解**
  - 一个关系模式的分解有多种方式，但分解后产生的模式都应该与原来模式等价
  - 等价可以从三个角度进行考虑：分解具有无损连接性、分解要保持函数依赖、分解既要保持函数依赖又要具有无损连接性。
  - 分解算法将一个关系分解成满足BCNF或3NF、具有无损连接性和保持函数依赖。