设计篇
第五章 逻辑数据库设计

主讲：程思瑶
海量数据计算研究中心

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　　ER图->关系表
- 逻辑数据库设计的目标
  - 满足用户的完整性和安全性要求；
  - 动态关系至少具有第三规范形式，静态关系至少具有第一规范形式；
  - 能够在逻辑级上高效率地支持各种数据库事务的运行；
  - 存储空间利用率高

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

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

5.1形成初始关系数据库模式

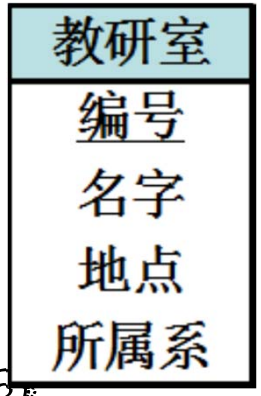
- 初始关系数据库模式是指直接由概念数据库模式生成的关系数据库模式。
- 初始关系数据库模式生成的目的是把概念数据库模式的**实体、实体间联系**等模型结构变换为**关系模式**。

5.1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

5.1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的码是S的主码。



教研室关系G(编号,名字, 地点,所属系)

5.1形成初始关系数据库模式

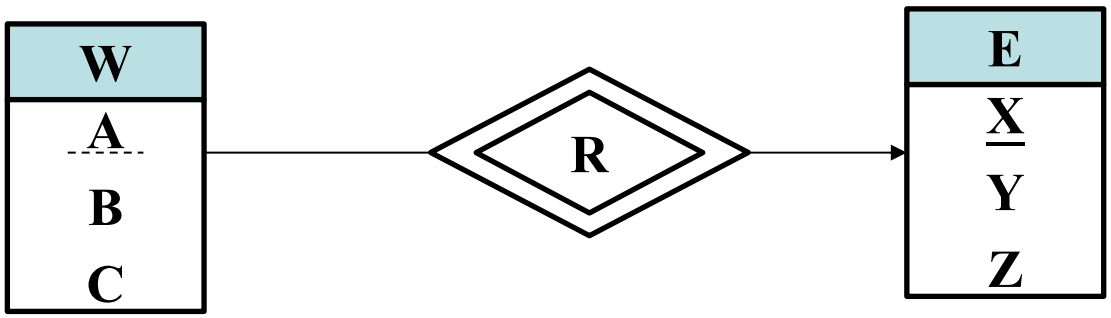
- 普通实体集的变换
  - 例

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

教师
身份证号
名字
工资
生日
职称
性别
地址
邮编
市
区
单位
学校
信箱

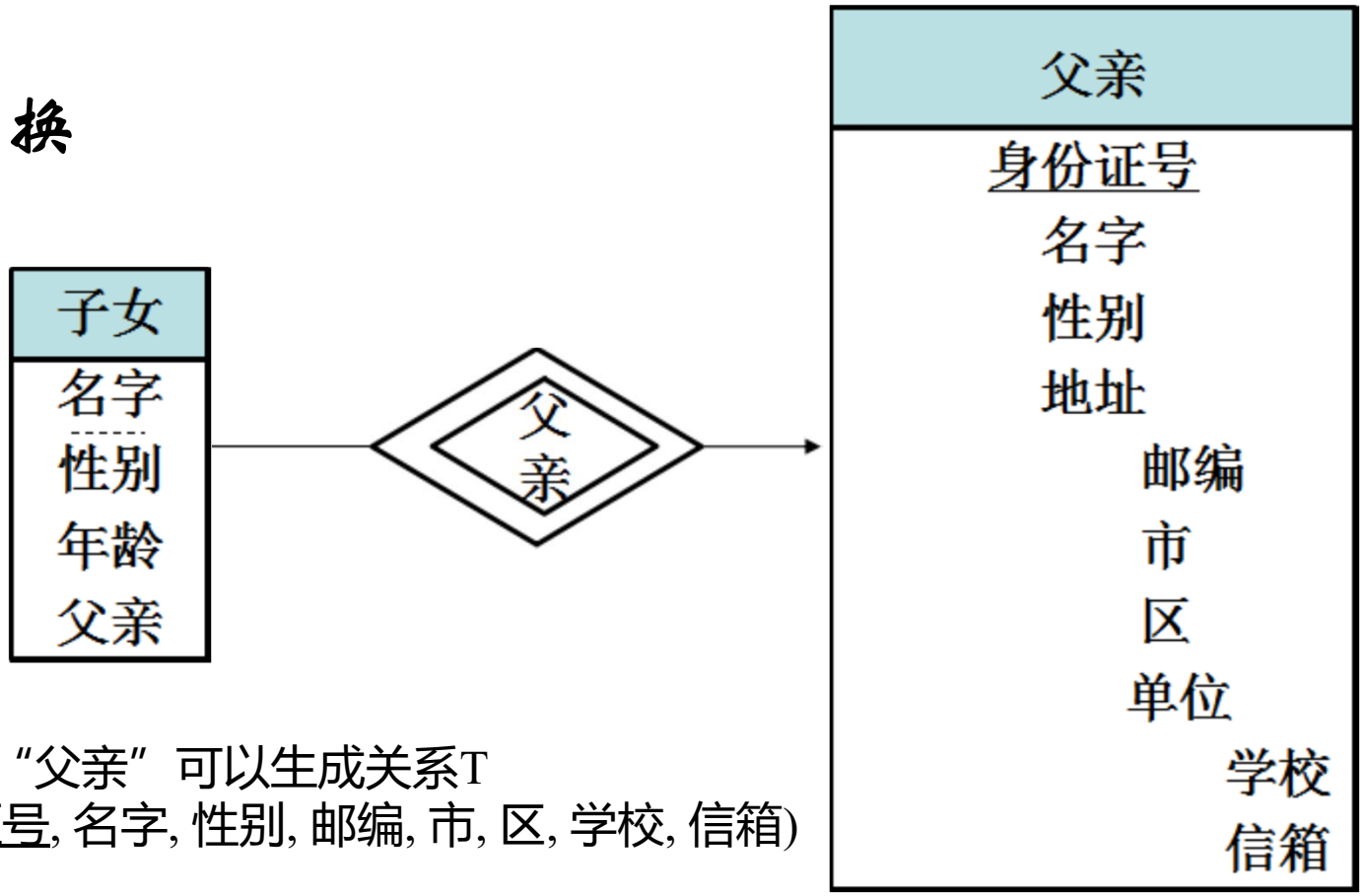
5.1形成初始关系数据库模式

- 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的码属性也是R的属性；
    - R的主码由E的码和W的部分码组合而成；
    - E对应的关系的码是R的外码。



5.1形成初始关系数据库模式

- 弱实体变换
  - 例



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

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

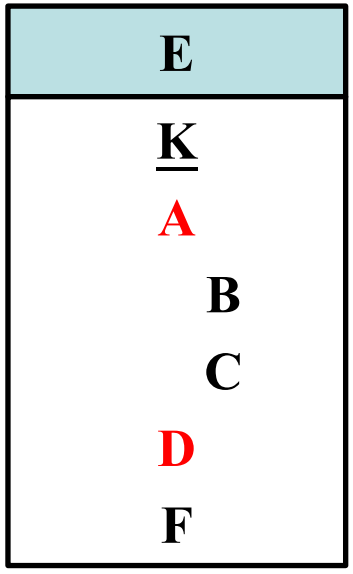
5.1形成初始关系数据库模式

- 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的主码K。A和K形成T的主码。
    - 如果A是复合属性，T包含A的简单子属性和S的码K。A的简单子属性和K形成T的码。
    - S关系中忽略属性A。
    - 对联系R的多值属性 类似处理

E
K
A
C
D

5.1形成初始关系数据库模式

- 多值属性的变换
  - 例



S(K,F)
T_1(K,B,C)
T2(K,D)

5.1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
    - 方法2：建立一个单独的关系W表示R
      - T和S的主码作为主码添入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。

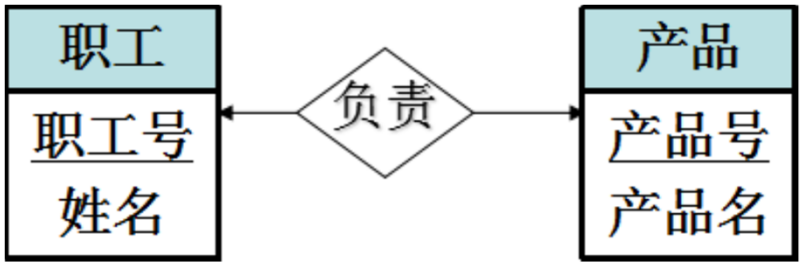
5.1形成初始关系数据库模式

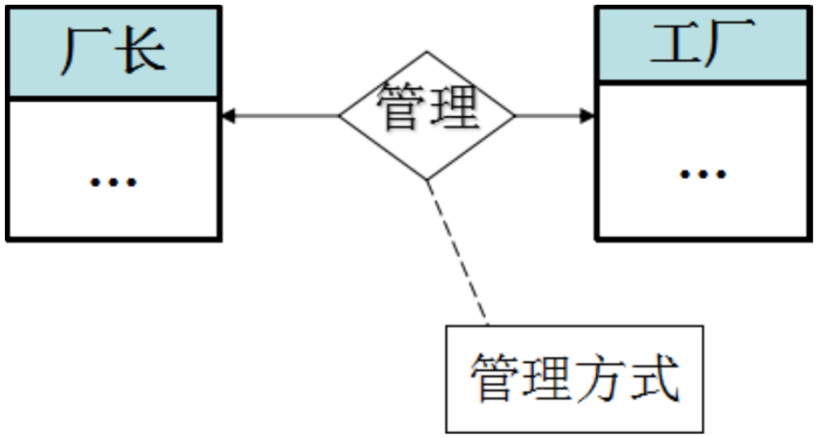
- 实体间联系的变换
  - 1:1联系的变换
    - 例

–职工(职工号，姓名)
–产品(产品号，产品名)
–负责(职工号，产品号)

–产品(产品号，产品名，职工号)
–职工(职工号，姓名)

–产品(产品号，产品名)
–职工(职工号，姓名，产品号)



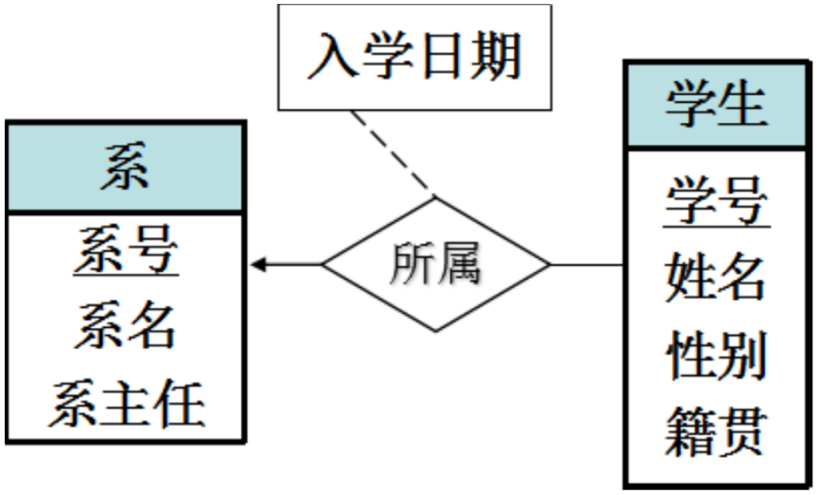


5.1形成初始关系数据库模式

- 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的主码作为外码添入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。
    - 方法2：建立一个单独的关系W表示R，同1:1联系。

5.1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
    - 例



–学生(学号，姓名，性别，籍贯)
–系(系号，系名，系主任)
–所属(学号，系号，入学日期)

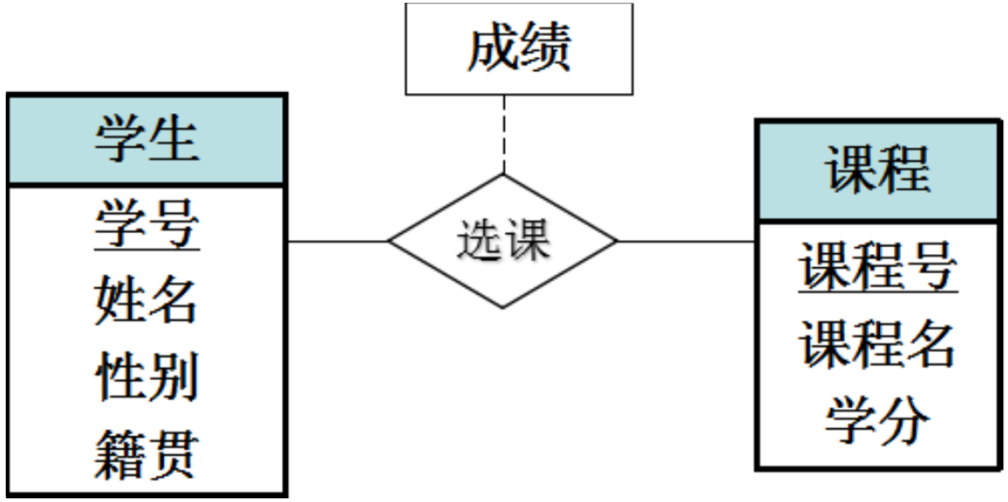
–学生(学号，姓名，性别，籍贯，系号，入学日期)
–系(系号，系名，系主任)

5.1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的主码添入W，既作为外码，也组合起来作为W的主码。
    - W还需要包含R的简单属性和复合属性的简单子属性。

5.1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
    - 例



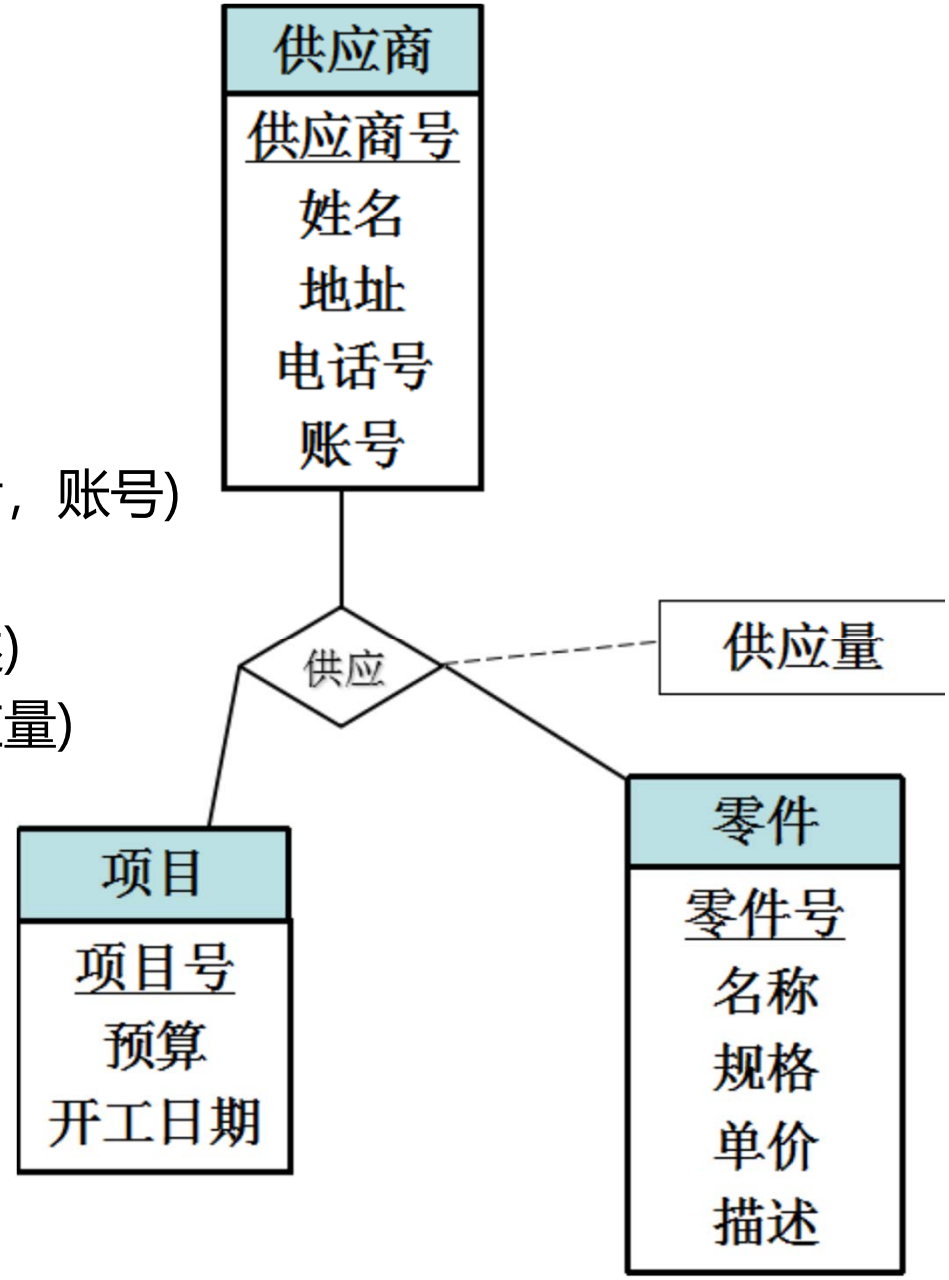
–学生(学号，姓名，性别，籍贯)
–课程(课程号，课程名，学分)
–选课(学号，课程号，成绩)

5.1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的主码都是T的外码，也组合起来作为T的主码。
      - 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

–供应商(供应商号，姓名，地址的，电话，账号)
–项目(项目号，预算，开工日期)
–零件(零件号，名称，规格，单价，描述)
–供应(供应商号，项目号，零件号，供应量)



5.1形成初始关系数据库模式

- 确定函数依赖集
  - 通过前面的步骤，初始关系数据库模式已经形成。
  - 最后，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；
  - 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。

5.2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冗余问题、插入问题、更新问题和删除问题。

5.2关系数据库设计理论

- 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（S#），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 ={S#, SD, MN, CN, G}

5.2关系数据库设计理论

- 例：

U ={S#, 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
- 得到属性集U上的函数依赖
  - F={S#→SD, SD→MN, (S#,CN)→G}

5.2关系数据库设计理论

- 例： U = {S#, 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**。
    - 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**。

5.2关系数据库设计理论

结论：Student关系模式不是一个好的模式。

- 一个“好”的模式应当不会发生插入异常、删除异常
- 更新异常、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，“姓名”→“年龄”这个函数依赖只有在没有同名人的条件下成立

相关概念

- 函数依赖
  - 如果X→Y而且Y不是X的子集，则称X→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→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) → Grade
平凡函数依赖：(Sno, Cno) → Sno
(Sno, Cno) → Cno

相关概念

- 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→Y，并且对于X的任何一个真子集Z，Z→Y都不成立，则称Y完全函数依赖于X。若X→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→X不成立，Z-X、Z-Y、Y-X不空。如果X→Y，Y→Z，则称Z传递地函数依赖于X。

- 例

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

Sno →Sname， Sno→Ssex， Sno → Sage， Sno → Sdept

(Sno, Sname) → Sdept,

(Sno, Ssex) → Sdept

例：在关系Std(Sno, Sdept, Mname)中，有：

Sno → Sdept，Sdept → Mname，

Mname传递函数依赖于Sno。

- 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的一个超码满足下列两个条件，且不存在K的真子集Z使得$Z \rightarrow U$。则称K是R的一个候选码：
    - 一个关系模式中可能具有多个候选码，指定其中的一个作为识别关系元组的主码(主键)。
    - 包含在任何一个候选码中的属性称为键属性。
    - 不包含在任何候选码中的属性称为非键属性。
    - 在最简单的情况下，候选码只包含一个属性。
    - 在最复杂的情况下，候选码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主码，则称X是R的外码。

函数依赖的公理系统

- 函数依赖的公理系统
  - 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公理系统。
  - Armstrong公理系统就是这样一个系统。

函数依赖的公理系统

- 函数依赖的公理系统
  - 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→Y都成立，则称F逻辑地蕴含X→Y。
  - 例：
    - 给定关系模式R=(A,B,C,G,H,I)及函数依赖F={A→B, A→C, CG→H, CG→I, B→H}, 则

      函数依赖A→H被F逻辑蕴含

函数依赖的公理系统

- Armstrong公理系统
  - 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

函数依赖的公理系统

- Armstrong公理系统
  - 设R是一个具有属性集合U的关系模式，F是R的一个函数依赖集合。Armstrong公理系统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。

Armstrong推理规则的正确性

(1) 自反律

设$Y \subseteq X \subseteq U$。

对$R<U, F>$的任一关系r中的任意两个元组t，s：

若$t[X]=s[X]$，由于$Y \subseteq X$，有$t[Y]=s[Y]$，

所以$X \rightarrow Y$成立.

函数依赖的公理系统

(2)增广律

设$X \to Y$为F所蕴含，且$Z \subseteq U$。

设$R<U, F>$的任一关系r中任意的两个元组t，s；

若$t[XZ]=s[XZ]$，则有$t[X]=s[X]$和$t[Z]=s[Z]$；

由$X \to Y$，于是有$t[Y]=s[Y]$，所以$t[YZ]=s[YZ]$，所以$XZ \to YZ$为F所蕴含.

函数依赖的公理系统

(3) 传递律

设$X \to Y$及$Y \to Z$为F所蕴含。

对$R<U, F>$的任一关系r中的任意两个元组t，s。

若$t[X]=s[X]$，由于$X \to Y$，有$t[Y]=s[Y]$；

再由$Y \to Z$，有$t[Z]=s[Z]$，所以$X \to Z$为F所蕴含.

函数依赖的公理系统

- 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$X \to Y$，$X \to Z$，有$X \to YZ$。

  （A2，A3）$X \to YX$，$XY \to ZY$

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

  （A2，A3）$XW \to YW \to Z$

- 分解规则：由$X \to Y$及$Z \subseteq Y$，有$X \to Z$。

  （A1，A3）$Z \subseteq Y$，$Y \to Z$

函数依赖的公理系统

- 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 $X \to A_1 A_2 \cdots A_k$成立的充分必要条件是$X \to A_i$成立（$i=1, 2, \cdots, k$）

函数依赖的公理系统

- 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→B, A→C, CG→H, CG→I, B→H}, 则

        A→H, CG→HI, AG →I,……

    - 设F为属性集U上的一组函数依赖，$X \subseteq U$，$X_F^+ = \{A | X \rightarrow A$能由$F$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。

函数依赖的公理系统

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

  - (1) 若$X=A_1$, $X^+ =\{A_1, A_2, A_3\}$
  - (2) 若$X=A_2$, $X^+ =\{A_2, A_3\}$
  - (3) 若$X=A_3$, $X^+ =\{A_3\}$

函数依赖的公理系统

- 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^+
  - （1）令$X^{(0)}=X$，$i=0$
  - （2）求B，这里$B=\{A \mid (\exists V)(\exists W)(V \to W \in F \wedge V \subseteq X^{(i)} \wedge A \in W)\}$；
  - （3）$X^{(i+1)}=B \cup X^{(i)}$
  - （4）判断$X^{(i+1)}=X^{(i)}$吗？
  - （5）若相等或$X^{(i)}=U$，则$X^{(i)}$就是X_F^+，算法终止。
  - （6）若否，则$i=i+1$，返回第（2）步。

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

算法至多执行|U-X|次循环

函数依赖的公理系统

- 例
  - U ={A,B,C,D,E},
  - F ={AB→C, AC→B，B→D，C→E，EC→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}∪{CD}={$ABCD$}。

函数依赖的公理系统

(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→C，B→D，C→E，AC→B，

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

(3) 因为$X^{(2)}=U$，算法终止

所以$(AB)_F^+ =\{ABCDE\}$。

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

函数依赖的公理系统

例. 设R<U,F>,其中U={A,B,C,D,E,I};

F={A→D,AB→E,BI→E,CD→I,E→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→D,E→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→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$(AE)_F^+=\{ACDEI\}$.

关系的候选码?

函数依赖的公理系统

- 快速求解候选码的方法

  对于给定的关系模式$R(A_1,A_2,\cdots,A_n)$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

函数依赖的公理系统

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

例：设R(A,B,C,D),F={D→B,B→D,AD→B,AC→D}

求R的候选码

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

所以，AC是候选码。

函数依赖的公理系统

定理2：对于给定的关系模式R及其F，如果X(X∈R)是R类属性，则X不在任何候选码中。

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的任一候选码中。

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的候选码。

函数依赖的公理系统

例：设R(A,B,C,D,E,P),

F={A→D,E→D,D→B,BC→D，DC→A},

求R的候选码。

解：C,E是L类属性，P是N类属性。

$(CEP)_F^+$=ABCDEP。所以CEP是候选码(键)。

函数依赖的公理系统

- 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。亦称为极小覆盖：
    - 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→A，使得F与F-{X→A}等价
    - F中不存在这样的函数依赖X→A，X有真子集Z使得F-{X→A}∪{Z→A}与F等价
- 极小函数依赖集不唯一
  - F={A→B，B→A，B→C，A→C，C→A}的极小函数依赖集为

    F1={A→B，B→C，C→A}，或

    F2={A→B，B→A，A→C，C→A}

函数依赖的公理系统

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→SDEPT，SDEPT→MN，（SNO，CNAME）→G}

F是极小覆盖么？

是!

(2) F' ={SNO→SDEPT，SNO→MN，SDEPT→MN，
(SNO，CNAME)→G，(SNO，SDEPT)→SDEPT}

F' 是极小覆盖么？

不是! 因为：F' -{SNO→MN}与F' 等价
F' -{(SNO，SDEPT)→SDEPT}也与F' 等价

函数依赖的公理系统

- 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$Y=A_1A_2\cdots A_k$，$k\geq 2$，则用$\{X\rightarrow A_j | j=1,2,\cdots,k\}$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$X\rightarrow A\in X_G^+$，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$X=B_1B_2\cdots B_m$，逐一考查B_i（$i=1,2,\cdots,m$），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几种
  - 第一范式（1NF）
  - 第二范式（2NF）
  - 第三范式（3NF）
  - BC范式（BCNF）
  - 多值依
  - ……

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

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

- 某一关系模式R为第n范式，可简记为 R∈nNF

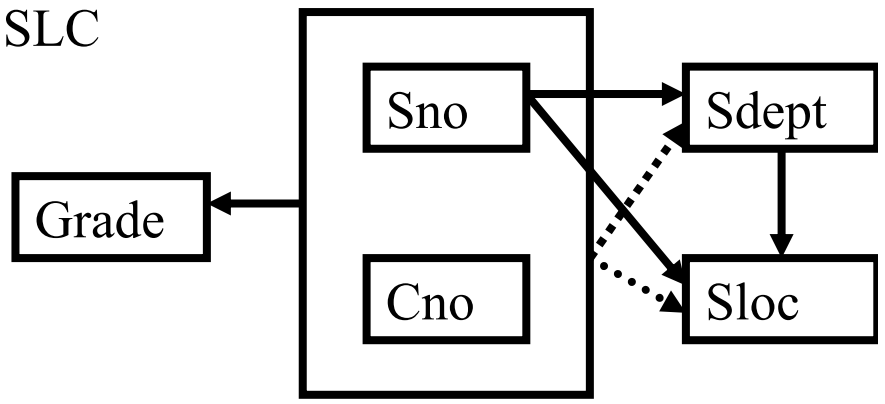
关系模式的规范形式

- 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。

关系模式的规范形式

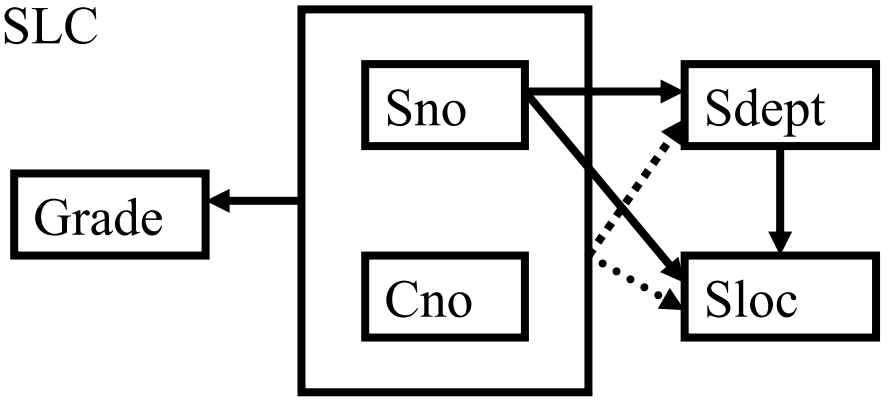
- 例，关系模式 SLC(Sno, Sdept, Sloc, Cno, Grade)，Sloc为学生住处，假设每个系的学生住在同一个地方。
- 函数依赖包括：

  (Sno, Cno)→Grade
  Sno → Sdept
  (Sno, Cno) → Sdept
  Sno → Sloc
  (Sno, Cno) → Sloc
  Sdept → Sloc



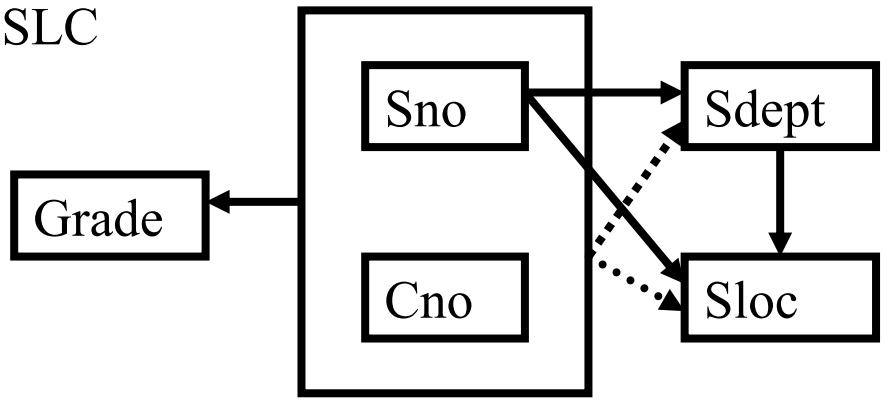
SLC的候选码为(Sno, Cno)

SLC满足第一范式
非键属性Sdept和Sloc部分函数依赖于候选码(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，还可能需要修改住处（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！



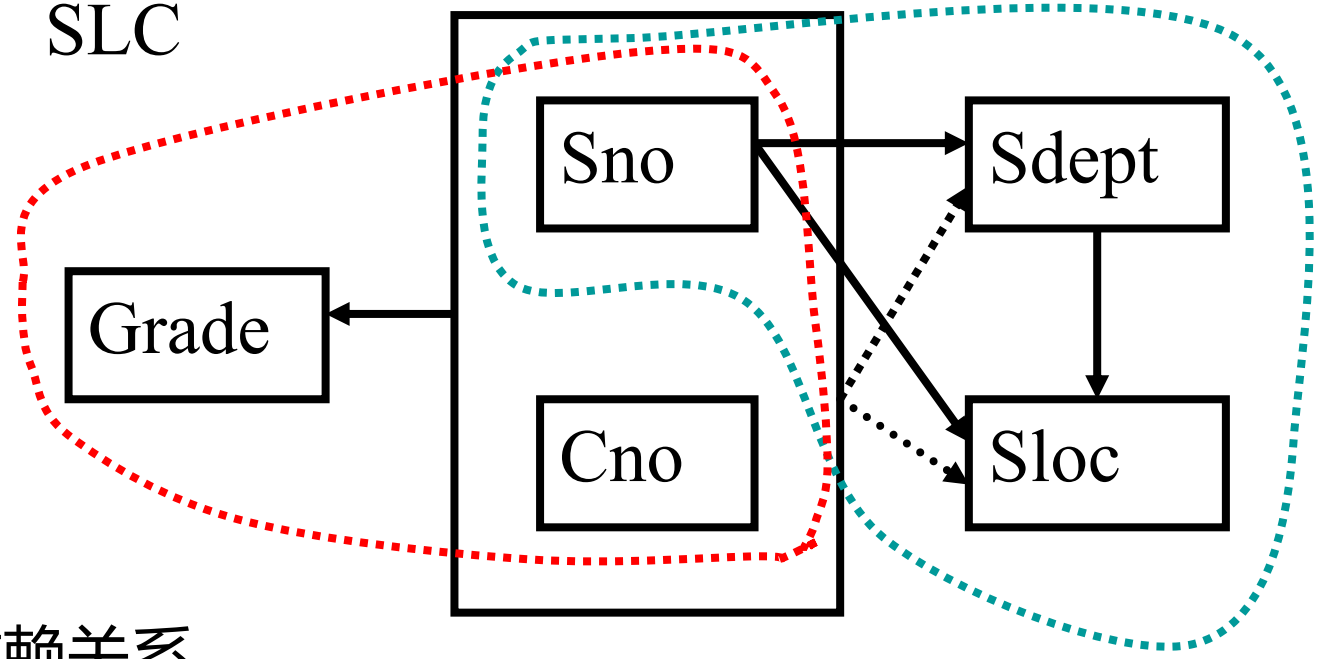
- 问题产生原因
  - Sdept、 Sloc部分函数依赖于候选码(Sno, Cno)

- 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）

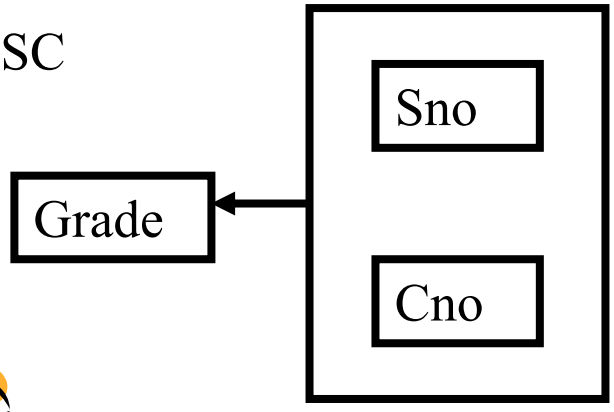
SL（Sno， Sdept， Sloc）

SLC

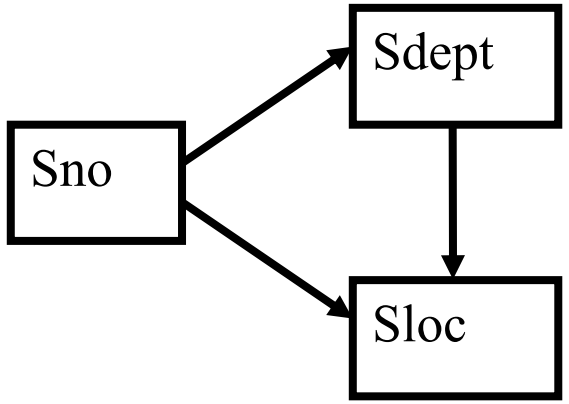


函数依赖关系

SC



SL



使上述四个问题在一定程度上得到了解决

关系模式的规范形式

- 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的候选码，则R称为第二范式关系模式，记作2NF。
  - 例如，

    SLC(Sno, Sdept, Sloc, Cno, Grade) ∈1NF

    SC（Sno， Cno， Grade）∈2NF

    SL（Sno， Sdept， Sloc）∈2NF

关系模式的规范形式

- 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**。

关系模式的规范形式

- 例， 2NF关系模式SL(Sno, Sdept, Sloc)中

  函数依赖： Sno → Sdept， Sdept → Sloc， Sno → Sloc

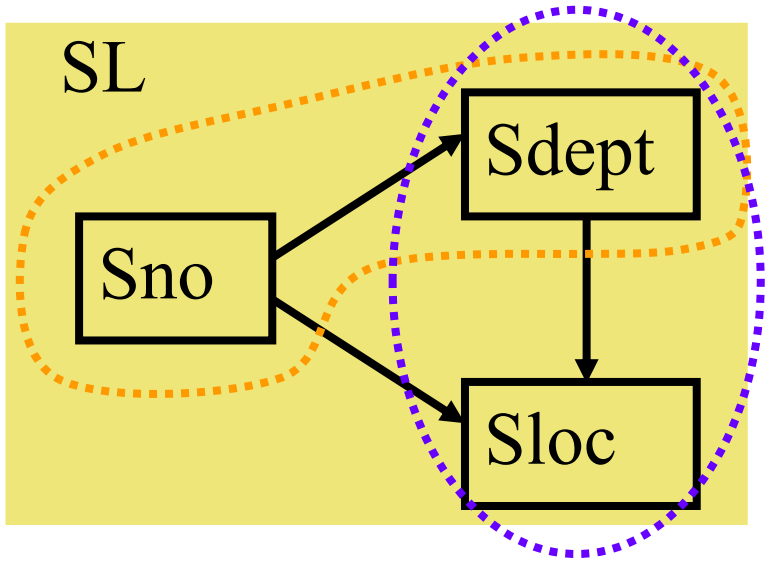
  Sloc传递函数依赖于Sno，即SL中存在非键属性对候选码的传递函数依赖。

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
  - 当学校调整系的地址时，由于关于每个系的地址信息是重复存储的，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
  - 把Sl

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

SD（Sno， Sdept）

DL（Sdept， Sloc）

SD的候选码为Sno， DL的候选码为Sdept

关系模式的规范形式

- 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码，则R称为第三范式关系模式，记作3NF。
  - 例：

SL(Sno, Sdept, Sloc) ∈ 2NF
SD（Sno， Sdept） ∈ 3NF
DL（Sdept， Sloc） ∈ 3NF

学生(学号，姓名，宿舍楼，宿舍号) ∈ 3NF

关系模式的规范形式

- 如果R∈3NF，则R也是2NF
- 若R∈3NF，则R的每一个非键属性既不部分函数依赖于候选码也不传递函数依赖于候选码。
- 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。

关系模式的规范形式

- 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)都可以作为候选码

    STJ∈3NF

关系模式的规范形式

- 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。

关系模式的规范形式

- 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。

因此虽然STJ∈3NF，但它仍不是一个理想的关系模式。

关系模式的规范形式

- 问题出现原因
  - 键属性J依赖于T，即键属性J部分依赖于候选码(S, T)。

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

- 解决方法
  - 将STJ分解为二个关系模式：

    SJ(S，J)

    TJ(T，J)

    SJ的候选码为(S,J),TJ的候选码为T(每个老师只教1门课)

关系模式的规范形式

- 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。
  - 定义：
    - 设关系模式R是1NF。如果对于R的每个函数依赖X→Y，则X必为候选码，则R是BCNF范式。

关系模式的规范形式

- 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}→P和{J,P}→S。

  {S,J}与{J,P}都是候选码。

  这个关系模式中显然没有属性对候选码的传递依赖或部分依赖。

  SPJ是3NF，同时也是BCNF。

关系模式的规范形式

- 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码；
  - 所有键属性都完全函数依赖于每个不包含它的候选码；
  - 没有任何属性函数依赖于非键的任何一组属性。

关系模式的规范形式

- 如果关系模式R∈BCNF，必定有R∈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。

关系模式的规范形式

- 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

关系模式的规范形式

- 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

↓ 消除复合属性、多值属性
1NF
↓ 消除非键属性对候选码的部分函数依赖
2NF
↓ 消除非键属性对候选码的传递函数依赖
3NF
↓ 消除键属性对候选码的部分和传递函数依赖
BCNF

关系模式的规范形式

- 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。

关系模式分解的约束

- 将一个关系模式R<U,F>分解为若干个关系模式$R_1<U_1,F_1>$，$R_2<U_2,F_2>$，…，$R_n<U_n,F_n>$（其中$U=U_1 \cup U_2 \cup \cdots \cup U_n$，且不存在$U_i \subseteq U_j$，$F_i$为F在$U_i$上的投影）
- 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，…，T_n中去（其中T_i是T在属性集U_i上的投影）。

关系模式分解的约束

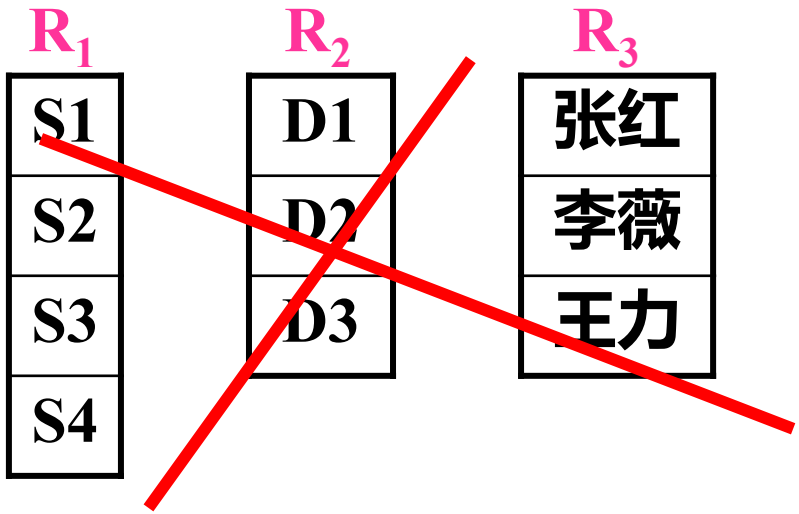
- 例，关系模式R的属性集合U={S#,SD,MN}，函数依赖集合F={S#→SD,SD→MN}。
- R的关系实例：

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？

关系模式分解的约束

- 例，关系模式R的属性集合U={S#,SD,MN}，函数依赖集合F={S#→SD,SD→MN}。
  - 分解1：ρ1={R1(S#)，R2(SD)，R3(MN)}



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。

关系模式分解的约束

- 例，关系模式R的属性集合U={S#,SD,MN}，函数依赖集合F={S#→SD,SD→MN}。
  - 分解2： ρ2 = {R1(S#，SD)，R2(S#，MN)}

R_1

S1	D1
S2	D1
S3	D2
S4	D3

R_2

S1	张红
S2	张红
S3	李薇
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→MN丢失了。

关系模式分解的约束

- 例，关系模式R的属性集合U={S#,SD,MN}, 函数依赖集合F={S#→SD,SD→MN}。
  - 分解3：ρ3 = {R1(S#,SD)，R2(SD,MN)}

R1

S1	D1
S2	D1
S3	D2
S4	D3

R2

D1	张红
D2	李薇
D3	王力

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。

关系模式分解的约束

- 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

关系模式分解的约束

- 具有无损连接性的模式分解
  - 设关系模式R<U,F>被分解为若干个关系模式 $R_1<U_1,F_1>$，$R_2<U_2,F_2>$，…，$R_n<U_n,F_n>$（其中$U=U_1 \cup U_2 \cup \cdots \cup U_n$，且不存在$U_i \subseteq U_j$，$F_i$为F在$U_i$上的投影），若R与$R_1$、$R_2$、…、$R_n$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。

关系模式分解的约束

- 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

  分解既要保持函数依赖，又要具有无损连接性

- **如何判断一个分解的无损连接性**

[例]：U={A,B,C,D,E},

F={A→C, B→C, C→D,DE→C ,CE→A}

ρ ={(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→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

B→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_{32}	b_{13}	b_{54}	a_5

C→D

	A	B	C	D	E
AD	a_1	b_{12}	b_{13}	a_4	b_{15}
AB	a_1	a_2	b_{13}	a_4	b_{25}
BE	b_{31}	a_2	b_{13}	a_4	a_5
CDE	b_{41}	b_{42}	a_3	a_4	a_5
AE	a_1	b_{32}	b_{13}	a_4	a_5

[例]：U={A,B,C,D,E},

F={A→C, B→C, C→D,DE→C ,CE→A}

ρ ={(A, D), (A, B), (B, E), (C, D, E), (A, E)}

C→D

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

DE→C

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

CE→A

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

ρ具有无损连接性

关系模式分解的约束

例： U={A,B,C,D,E}, F={AB→C, C→D,D→E}

ρ ={(A, B, C), (C, D), (D, E)}

试问ρ是否具有无损连接性？