第5章：逻辑数据库设计

Logical Database Design

邹兆年

哈尔滨工业大学
计算机科学与技术学院
海量数据计算研究中心
电子邮件：znzou@hit.edu.cn

2019年春

教学内容[1]

1. ER模型转换为关系数据库模式
2. 函数依赖
3. 关系模式的范式
4. 关系模式分解

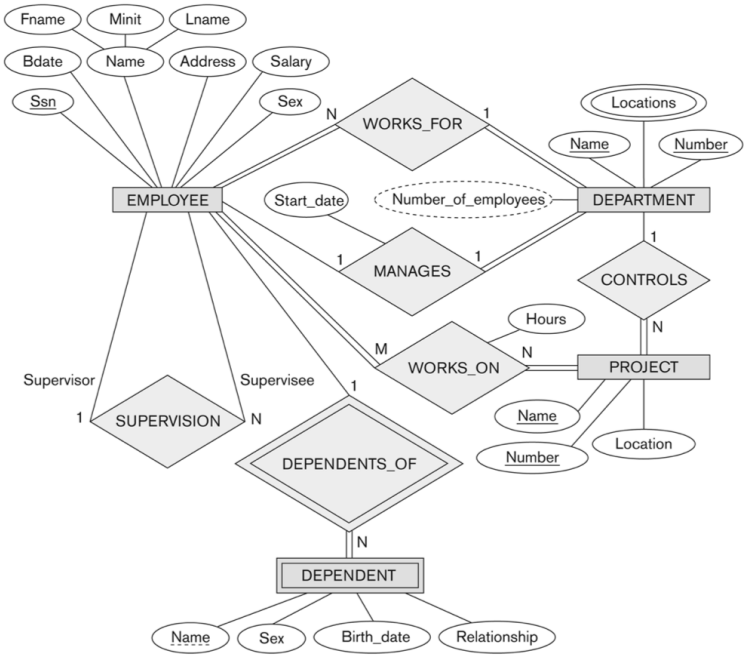
[1]课件更新于2019年3月27日

5.1 ER模型转换为关系数据库模式

From the ER model to Relational Database Schemas

ER模型转换为关系数据库模式

- 在逻辑数据库设计阶段，首先要将ER模型转换为关系数据库模式
  - 与实体相关的概念 $\longrightarrow$ 关系模式
  - 与联系相关的概念 $\longrightarrow$ 关系模式
- 例: 第4章中COMPANY数据库的ER图

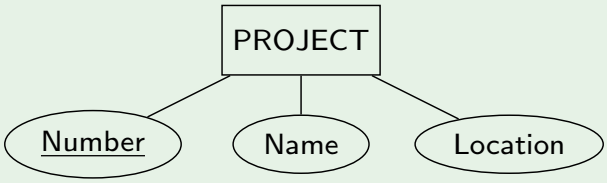


实体型的转换

转换规则

1. 实体型的名称$\longrightarrow$关系名
2. 实体型的属性集$\longrightarrow$关系的属性集(多值属性除外)
3. 实体型的主键$\longrightarrow$关系的主键
4. 实体$\longrightarrow$元组

Example (实体型的转换)



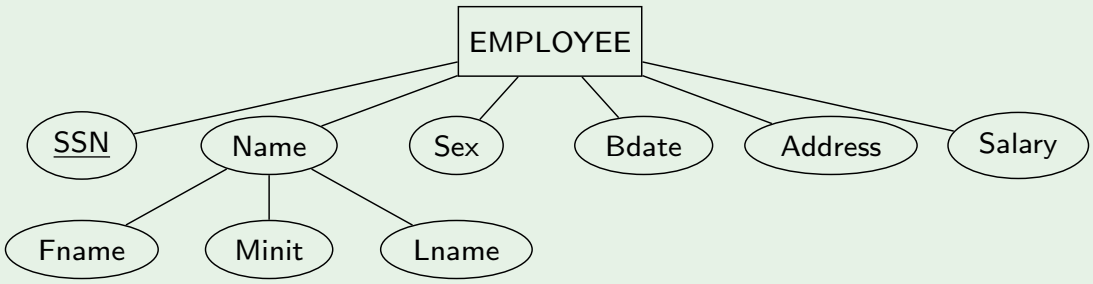
PROJECT(Number, Name, Location)

复合属性的转换

转换规则

1. 复合属性的最低级组成成分$\longrightarrow$关系的属性

Example (复合属性的转换)



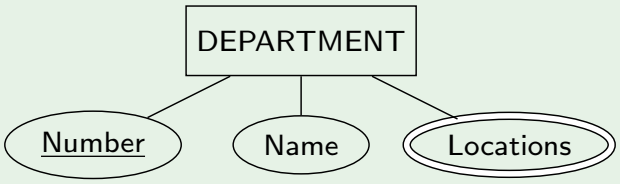
EMPLOYEE(SSN, Fname, Minit, Lname, Sex, Bdate, Address, Salary)

多值属性的转换

转换规则

1. 实体型关系的属性集中不包含多值属性
2. 多值属性$\longrightarrow$关系
3. 将实体型的主键并入多值属性关系中
4. 建立多值属性关系到实体型关系的外键约束

Example (多值属性的转换)



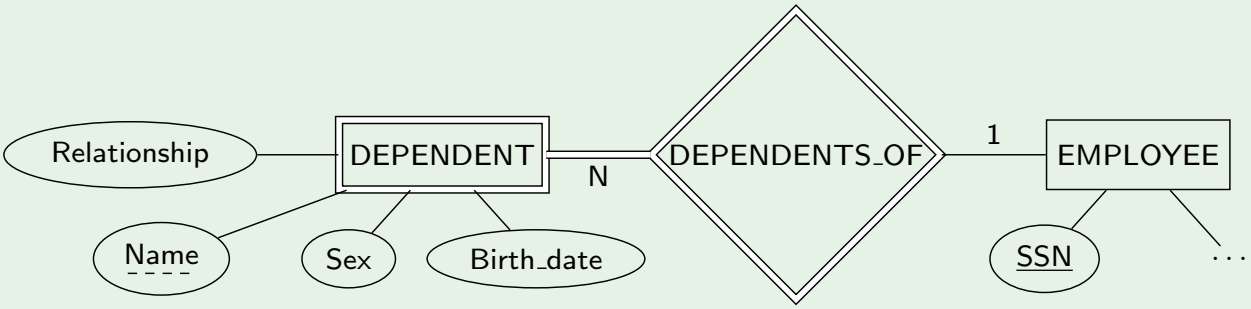
DEPARTMENT(Number, Name)
Locations(Number, Location)

弱实体型的转换

转换规则

1. 弱实体型的名称$\longrightarrow$关系名
2. 弱实体型的属性集$\cup$属主实体型的主键$\longrightarrow$关系的属性集
3. 属主实体型的主键$\cup$弱实体型的部分键$\longrightarrow$关系的主键
4. 建立弱实体型关系到属主实体型关系的外键约束

Example (弱实体型的转换)



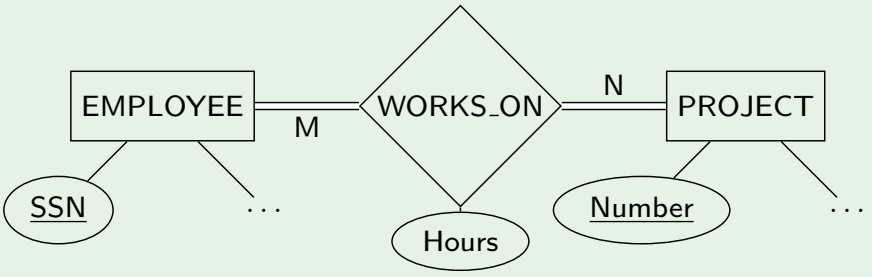
DEPENDENT(SSN, Name, Sex, Birth_date, Relationship)

M:N二元联系型的转换

转换规则

1. 联系型的名称$\longrightarrow$关系名
2. 实体型1的主键$\cup$实体型2的主键$\cup$联系型的属性集$\longrightarrow$关系的属性集
3. 实体型1的主键$\cup$实体型2的主键$\longrightarrow$关系的主键
4. 建立联系型关系到实体型1关系的外键约束
5. 建立联系型关系到实体型2关系的外键约束

Example (M:N二元联系型的转换)



WORKS_ON(SSN, Number, Hours)

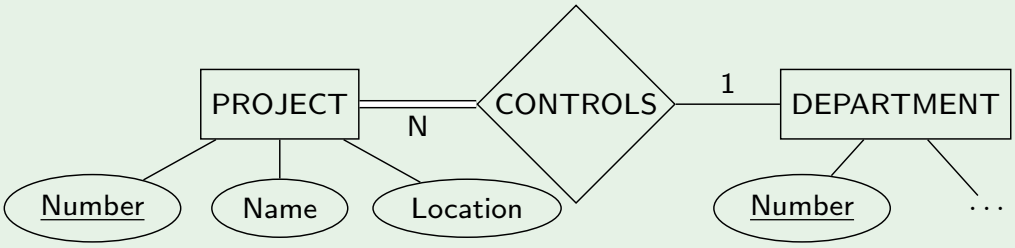
N:1二元联系型的转换

转换规则

设联系型中基数N一方的实体型(如EMPLOYEE)为实体型1，基数1一方的实体型(如DEPARTMENT)为实体型2

1. 将实体型2的主键和联系型的属性并入实体型1的属性集中
2. 建立实体型1关系到实体型2关系的外键约束

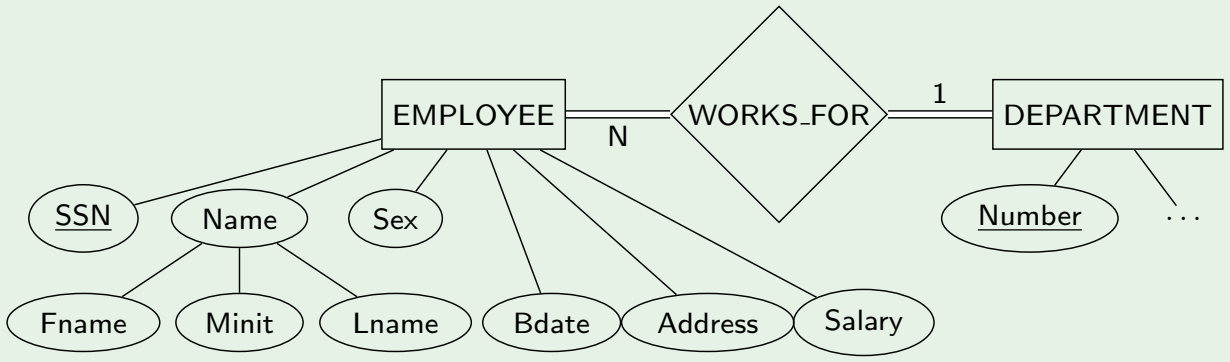
Example (N:1二元联系型的转换)



PROJECT(Number, Name, Location, DNumber)，其中Dnumber参照DEPARMENT.Number

N:1二元联系型的转换(续)

Example (N:1二元联系型的转换)



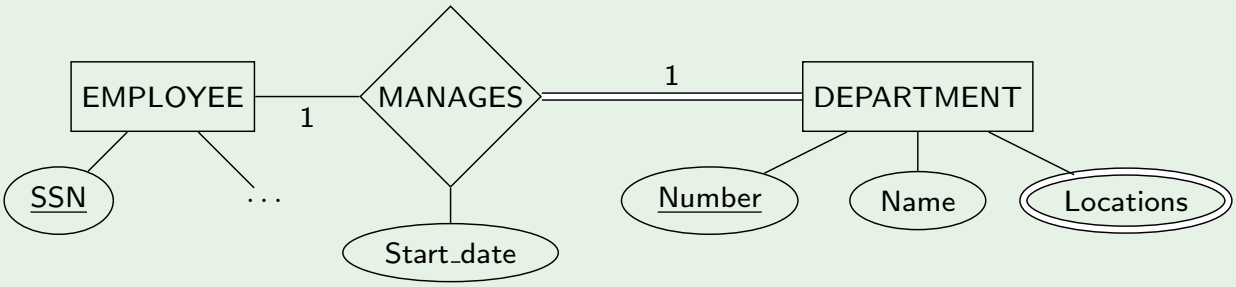
EMPLOYEE(SSN, Fname, Minit, Lname, Sex, Bdate, Address, Salary, Number)，其中Number参照DEPARMENT.Number

1:1二元联系型的转换

转换规则

与N:1联系的转换方法相同

Example (1:1二元联系型的转换)



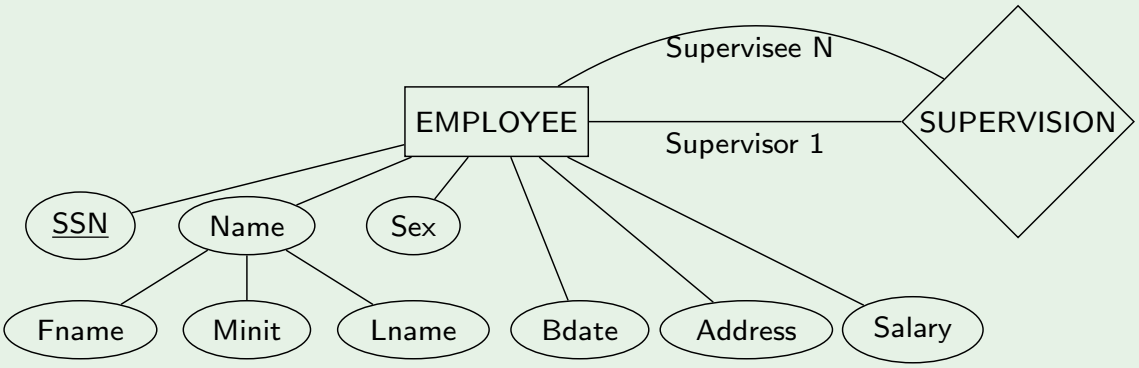
DEPARTMENT(Number, Name, SSN, Start_date)，其中DEPARMENT.SSN参照EMPLOYEE.SSN

二元自联系型的转换

转换规则

1. 与不同实体型间二元联系型的转换方法相同
2. 对重名属性重命名

Example (二元自联系型的转换)

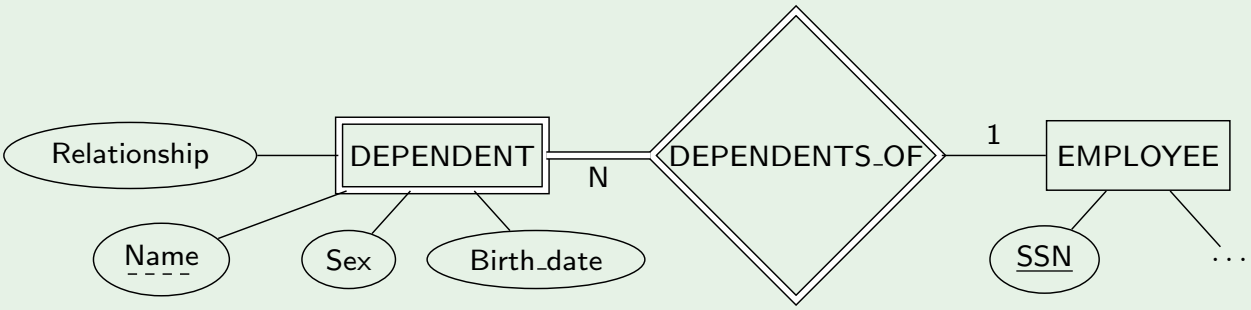


EMPLOYEE(SSN, Fname, Minit, Lname, Sex, Bdate, Address, Salary, SupervisorSSN)，其中SupervisorSSN参照EMPLOYEE.SSN

标识联系型的转换

- 由于弱实体型转换成的关系已经包含了属主实体型的主键，因此标识联系型就不必转换了

Example (标识联系型的转换)



DEPENDENT(SSN, Name, Sex, Birth_date, Relationship)

ER模型转换为关系数据库模式

Example (COMPANY关系数据库模式)

COMPANY数据库的ER模型转换为如下关系数据库模式

- 实体型和N:1联系型转换成的关系模式
  - EMPLOYEE(SSN, Fname, Minit, Lname, Sex, Bdate, Address, Salary, Number, SupervisorSSN)
  - PROJECT(Number, Name, Location, DNumber)
  - DEPARTMENT(Number, Name, SSN, Start_date)
  - DEPENDENT(SSN, Name, Sex, Birth_date, Relationship)
- 多值属性转换成的关系模式
  - Locations(Number, Location)
- M:N联系型转换成的关系模式
  - WORKS_ON(SSN, Number, Hours)
- 若没有概念数据库设计，很难保证设计出这样的关系数据库模式

下一步该做什么?

关系数据库模式的评估

- 目的: 评价一个关系数据库模式设计的"好坏"
- 方法: 可以用关系数据库规范化理论(第5.3节)来评估关系数据库模式设计的"好坏"(即规范化程度)
  - 关系模式越规范，数据更新时产生的问题越少，因此就越"好"
  - 关系模式越不规范，数据更新时产生的问题越多，因此就越"差"

重要提示

- 关系数据库规范化程度不是评估关系数据库模式的唯一准则
- 某些情况下，规范的关系模式在数据库系统中的实际表现并不好
- 要"因地制宜"

5.2 函数依赖

Functional Dependencies

数据依赖(Data Dependency)

- 数据依赖(data dependency): 关系的属性在语义上的依赖关系

Example (数据依赖)

关系模式SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)[a]的属性间存在如下数据依赖:

- 一个学号(Sno)只对应一个学生姓名(Sname)
- 一名学生(Sno)只属于一个系(Sdept)
- 一个系(Sdept)只有一名系主任(Sdean)
- 一名学生(Sno)只对应一名系主任(Sdean)
- 一名学生学习一门课程(Sno, Cno)只有一个成绩(Grade)

[a]SDC代表Student-Department-Course

- 关系模式中的不合理数据依赖会导致一系列问题
  - 数据更新问题: 数据插入异常、数据删除异常、数据修改繁琐
  - 数据冗余

问题1: 数据插入异常

- 现象: 该插入的数据无法插入

Example (数据插入异常)

如果一个系刚成立，尚无学生，则无法把这个系及其系主任的信息存入数据库，因为SDC关系的主键是(Sno, Cno)，该插入操作违反了实体完整性约束

Sno	Sname	Sdept	Sdean	Cno	Grade
221101	Nick	Physics	Einstein	1002	92
221101	Nick	Physics	Einstein	1003	85
221101	Nick	Physics	Einstein	1006	88
231101	Elsa	CS	Turing	1006	90
231101	Elsa	CS	Turing	1003	95
231102	Eric	CS	Turing	1003	80
232101	Abby	Math	Gauss	1002	100
		AI	Minsky		

问题2: 数据删除异常

- 现象: 不该删除的数据不得不删除

Example (数据删除异常)

如果某系的学生全部毕业了，那么在删除该系全体学生信息的同时，该系及其系主任的信息也被删除了

Sno	Sname	Sdept	Sdean	Cno	Grade
221101	Nick	Physics	Einstein	1002	92
221101	Nick	Physics	Einstein	1003	85
221101	Nick	Physics	Einstein	1006	88
~~231101~~	~~Elsa~~	~~CS~~	~~Turing~~	~~1006~~	~~90~~
~~231101~~	~~Elsa~~	~~CS~~	~~Turing~~	~~1003~~	~~95~~
~~231102~~	~~Eric~~	~~CS~~	~~Turing~~	~~1003~~	~~80~~
232101	Abby	Math	Gauss	1002	100

问题3: 数据修改繁琐

- 现象: 数据修改非常繁琐，容易出错

Example (数据修改繁琐)

某系更换系主任后，必须修改该系全体学生的元组，否则会导致数据不一致

Sno	Sname	Sdept	Sdean	Cno	Grade
221101	Nick	Physics	Einstein	1002	92
221101	Nick	Physics	Einstein	1003	85
221101	Nick	Physics	Einstein	1006	88
231101	Elsa	CS	~~Turing~~ Shannon	1006	90
231101	Elsa	CS	~~Turing~~ Shannon	1003	95
231102	Eric	CS	~~Turing~~ Shannon	1003	80
232101	Abby	Math	Gauss	1002	100

问题4: 数据冗余

- 现象: 存在大量冗余数据

Example (数据冗余)

1. 一个学生的姓名和系可能重复出现: 该学生选了几门课，就重复出现几次
2. 系主任的姓名可能重复出现，重复次数与该系所有学生的选课元组数相同

Sno	Sname	Sdept	Sdean	Cno	Grade
221101	Nick	Physics	Einstein	1002	92
221101	Nick	Physics	Einstein	1003	85
221101	Nick	Physics	Einstein	1006	88
231101	Elsa	CS	Turing	1006	90
231101	Elsa	CS	Turing	1003	95
231102	Eric	CS	Turing	1003	80
232101	Abby	Math	Gauss	1002	100

问题成因及解决方法

- 结论：关系模式SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)设计得不好
- 原因：由某些不合适的数据依赖导致的
- 解决方法：
  1. 运用关系数据库规范化理论(第5.3节)来评估关系模式的规范化程度
  2. 通过关系模式分解(第5.4节)来消除不合适的数据依赖
- 本节介绍和数据依赖有关的概念和理论

函数依赖(Functional Dependency)的定义

Definition (函数依赖(functional dependency))

设$R(U)$是属性集U上的关系模式，$X \subseteq U$，$Y \subseteq U$。对于$R(U)$的任意关系实例中的任意两个元组t_1和t_2，如果由$t_1[X] = t_2[X]$可以推出$t_1[Y] = t_2[Y]$，则称X函数决定Y，或Y函数依赖于X，记作$X \to Y$。

Example (函数依赖)

关系模式SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)中存在如下函数依赖：

- Sno $\to$ Sname：一个学号只对应一个学生姓名
- Sno $\to$ Sdept：一名学生只属于一个系
- Sdept $\to$ Sdean：一个系只有一名系主任
- Sno $\to$ Sdean：一名学生只对应一名系主任
- (Sno, Cno) $\to$ Grade：一名学生学习一门课程只有一个成绩

函数依赖的定义(续)

- 如果Y不函数依赖于X，则记作$X \nrightarrow Y$
- 如果$X \rightarrow Y$且$Y \rightarrow X$，则记作$X \leftrightarrow Y$

重要提示

- 函数依赖是关系模式的所有关系实例上都成立的依赖关系，不能只根据某些关系实例来确定函数依赖
- 函数依赖是语义范畴的概念，只能根据数据的语义来确定函数依赖。例如，$Sname \rightarrow Sno$只有在学生不同名时才成立
- 数据库设计者可以对现实世界作出强制规定。例如，规定学生不允许同名，从而使得$Sname \rightarrow Sno$成立

函数依赖的类型

Definition (平凡(trivial)函数依赖)

若$Y \subseteq X$，则$X \rightarrow Y$是平凡函数依赖

Definition (非平凡(nontrivial)函数依赖)

若$Y \not\subseteq X$，则$X \rightarrow Y$是非平凡函数依赖

Definition (完全函数依赖(full functional dependency))

在关系模式$R(U)$中，如果$X \rightarrow Y$，且对任意$X' \subset X$，都有$X' \nrightarrow Y$，则称Y完全函数依赖于X，记作$X \xrightarrow{f} Y$

Example (完全函数依赖)

在关系模式SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)中，

- $(Sno, Cno) \xrightarrow{f} Grade$，因为$Sno \nrightarrow Grade$且$Cno \nrightarrow Grade$

函数依赖的类型(续)

Definition (部分函数依赖(partial functional dependency)))

在关系模式$R(U)$中，如果$X \to Y$，且存在$X' \subset X$，使得$X' \to Y$，则称Y部分函数依赖于X，记作$X \xrightarrow{p} Y$

Definition (传递函数依赖(transitive functional dependency)))

在关系模式$R(U)$中，如果$X \to Y$，$Y \to Z$，且$Y \not\subseteq X$，$Y \not\to X$，则称Z传递函数依赖于X，记作$X \xrightarrow{t} Z$

Example (传递函数依赖)

在关系模式SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)中，

- $(Sno, Cno) \xrightarrow{p} Sname$，因为$Sno \to Sname$
- $Sno \xrightarrow{t} Sdean$，因为$Sno \to Sdept$，$Sdept \not\to Sno$且$Sdept \to Sdean$

Question

使用函数依赖的概念重新定义候选键

函数依赖的公理系统

为什么要学习函数依赖的公理系统?

- 在设计关系模式时，数据库设计者只能给出一部分函数依赖，无法(也没必要)给出全部函数依赖
- 在使用关系数据库规范化理论评估关系模式的规范化程度时，仅知道数据库设计者明确给出函数依赖很可能是不够的
- 使用函数依赖的公理系统，可以在数据库设计者给出函数依赖的基础上推导出其他成立的函数依赖

Armstrong公理系统

Definition (逻辑蕴含)

设$R(U,F)$是一个关系模式，其属性集为U，函数依赖集为F。若在R的任意关系实例r中，函数依赖$X \to Y$都成立(即对于r中任意两个元组t_1和t_2，若$t_1[X] = t_2[X]$，则$t_1[Y] = t_2[Y]$)，则称F逻辑蕴含$X \to Y$，记作$F \models X \to Y$。

Armstrong公理系统

设$R(U,F)$是一个关系模式，Armstrong公理系统包含以下3条推理规则:

1. 自反律: 若$Y \subseteq X \subseteq U$，则$F \models X \to Y$
2. 增广律: 若$F \models X \to Y$，则对任意$Z \subseteq U$，有$F \models XZ \to YZ$[a]
3. 传递律: 若$F \models X \to Y$且$F \models Y \to Z$，则$F \models X \to Z$

[a]在关系数据库规范化理论中，XY代表$X \cup Y$

Armstrong公理系统(续)

Theorem

*Armstrong*公理系统是正确*(sound)*且完备*(complete)*的

- 正确性: 使用*Armstrong*公理系统推出的任何函数依赖一定被F逻辑蕴含
- 完备性: 任何被F逻辑蕴含的函数依赖一定能够使用*Armstrong*公理系统推出

Example (Armstrong公理系统)

关系模式SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)上的函数依赖集为$F = \{Sno \to Sname, Sno \to Sdept, Sdept \to Sdean, (Sno, Cno) \to Grade\}$，证明$F \models (Sno, Cno) \to (Sdean, Grade)$
(证明过程见黑板)

Armstrong公理系统(续)

Armstrong公理系统的导出规则

设$R(U,F)$是一个关系模式

1. 合并规则：若$F \models X \to Y$且$F \models X \to Z$，则$F \models X \to YZ$
2. 伪传递规则：若$F \models X \to Y$且$F \models WY \to Z$，则$F \models XW \to Z$
3. 分解规则：若$F \models X \to Y$，则对任意$Z \subseteq Y$，有$F \models X \to Z$

Question

证明3条导出规则成立

属性集的闭包(Closure)

为什么要学习属性集的闭包?

- 使用Armstrong公理系统可以证明某函数依赖是否被逻辑蕴含，但是不够方便
- 使用属性集的闭包可以更方便地证明某函数依赖是否被逻辑蕴含

Definition (属性集的闭包)

设$R(U,F)$是一个关系模式，集合$\{A|A \in U, F \models X \to A\}$称为$X$关于$F$的闭包(closure)，记作$X_F^+$

Theorem

设$R(U,F)$是一个关系模式，$X,Y \subseteq U$，$F \models X \to Y$的充要条件是$Y \subseteq X_F^+$

- 逻辑蕴含判定问题 $\Rightarrow$ 属性集闭包计算问题

属性集闭包的计算

算法: 计算属性集的闭包

输入: 属性集X、函数依赖集F
输出: X关于F的闭包X_F^+

1: $i \leftarrow 0$; $X^{(0)} \leftarrow X$ {初始化}
2: **repeat**
3: $B \leftarrow \{A | V \rightarrow W \in F, V \subseteq X^{(i)}, A \in W\}$ {匹配函数依赖}
4: $X^{(i+1)} \leftarrow X^{(i)} \cup B$ {扩充闭包}
5: $i \leftarrow i + 1$
6: **until** $X^{(i)} = X^{(i-1)}$或$X^{(i)} = U$ {终止条件}
7: **return** $X^{(i)}$

Example (属性集的闭包)

关系模式SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)上的函数依赖集为$F = \{Sno \rightarrow Sname, Sno \rightarrow Sdept, Sdept \rightarrow Sdean, (Sno, Cno) \rightarrow Grade\}$，证明$F \models (Sno, Cno) \rightarrow (Sdean, Grade)$

属性集闭包的计算(续)

Example (属性集的闭包)

1. 计算$(Sno, Cno)_F^+$

i	匹配的函数依赖	集合$X^{(i)}$
0		(Sno, Cno)
1	$Sno \rightarrow Sname$ $Sno \rightarrow Sdept$ $Sdept \rightarrow Sdean$ $(Sno, Cno) \rightarrow Grade$	$(Sno, Cno, Sname, Sdept, Grade)$
2	~~$Sno \rightarrow Sname$~~ ~~$Sno \rightarrow Sdept$~~ $Sdept \rightarrow Sdean$ ~~$(Sno, Cno) \rightarrow Grade$~~	$(Sno, Cno, Sname, Sdept, Grade, Sdean)$ $(X^{(i)} = U$，终止$)$

2. 因为$(Sdean, Grade) \subseteq (Sno, Cno)_F^+$，所以$F \models (Sno, Cno) \rightarrow (Sdean, Grade)$

等价函数依赖集

为什么要学习等价函数依赖集?

- 在设计关系模式时，不同设计者会给出(表面上)不同的函数依赖集
- 如果这些表面上不同的函数依赖集能够逻辑蕴含相同的函数依赖，那么它们本质上是一样的，因此是等价的
- 然而，等价的函数依赖集的简练程度却未必相同，我们希望从全部等价的函数依赖集中找出最简练的那个

等价函数依赖集的相关概念

Definition (函数依赖集的闭包)

设$R(U,F)$是一个关系模式，由F逻辑蕴含的全部函数依赖的集合叫做F的闭包(closure)，记作F^+

Definition (等价函数依赖集)

设F和G是关系模式$R(U)$上的两个函数依赖集，如果$F^+ = G^+$，则F与G等价

Definition (函数依赖集的覆盖)

设F和G是关系模式$R(U)$上的两个函数依赖集，如果$G^+ \subseteq F^+$ (即G中每个函数依赖都被F逻辑蕴含)，则称F覆盖G (F covers G)

Theorem

函数依赖集F与G等价，当且仅当F覆盖G且G覆盖F

等价函数依赖集的判定

- 方法1: 使用$F^+ = G^+$来判定F与G等价，但是计算F^+和G^+的代价太高
- 方法2: 如果F覆盖G(F逻辑蕴含G中任意函数依赖)，且G覆盖F(G逻辑蕴含F中任意函数依赖)，则F与G等价

Example (等价函数依赖集)

判断F和G是否等价:

- $F = \{Sno \to Sname, Sno \to Sdept, Sdept \to Sdean, (Sno, Cno) \to Grade\}$
- $G = \{Sno \to Sname, Sno \to Sdept, Sno \to Sdean, (Sno, Cno) \to Grade\}$

我们可以使用Armstrong公理或属性集闭包证明

- F逻辑蕴含G中所有函数依赖
- $G \not\vdash Sdept \to Sdean$

所以F和G不等价

最小覆盖(Canonical Cover)

Definition (最小覆盖)

关系模式$R(U)$的函数依赖集F的最小覆盖(canonical cover)是满足下列3个条件的F的等价函数依赖集F_c

1. (F_c中不存在冗余函数依赖) F_c中不存在函数依赖$X \to Y$，使得F_c与$F_c - \{X \to Y\}$等价
2. (F_c函数依赖左部不存在冗余属性) F_c中不存在函数依赖$X \to Y$，有$Z \subset X$使得F_c与$(F_c - \{X \to Y\}) \cup \{Z \to Y\}$等价
3. (F_c函数依赖左部唯一) F_c中任意两个函数依赖的左部不相同[a]

[a] 有些教材中使用另一种条件: F_c中任意函数依赖的右部只包含1个属性

最小覆盖的计算

最小覆盖的计算算法

输入: 函数依赖集F
输出: F的最小覆盖

1: **repeat**
2: (删除冗余函数依赖) 对F中任意函数依赖$X \to Y$，若F与$F - \{X \to Y\}$等价，则将$\{X \to Y\}$从F中删除
3: (删除函数依赖左部的冗余属性) 对F中任意函数依赖$X \to Y$，若存在$A \in X$，使得F与$F - \{X \to Y\} \cup \{X - \{A\} \to Y\}$等价，则将$X \to Y$替换为$X - \{A\} \to Y$
4: (合并函数依赖) 使用合并规则，将F中左部相同的两个函数依赖$X \to Y$和$X \to Z$合并为一个函数依赖$X \to YZ$
5: **until** F不再变化
6: **return** F

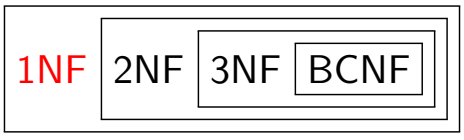
5.3 关系模式的范式

Normal Forms of Relation Schemas

关系模式的范式(Normal Forms)

- 范式(normal form): 在关系数据库规范化理论中，根据关系模式的规范化程度不同，把关系模式划分为若干类，称作范式
- 范式的级别
  - 第一范式(1NF)
  - 第二范式(2NF)
  - 第三范式(3NF)
  - Boyce-Codd范式(BC范式，BCNF)
  - 第四范式(4NF)
  - ...
- 范式的关系: $4NF \subseteq BCNF \subseteq 3NF \subseteq 2NF \subseteq 1NF$
- 关系模式的规范化程度越高，数据更新导致的问题越少
- 在函数依赖的范畴内，我们只讨论1NF、2NF、3NF和BCNF

关系模式的范式



第一范式(1NF)

Definition (第一范式(1NF))

如果关系模式R的每个属性都是不可分的，则称R为第一范式关系模式，记作$R \in 1NF$

Example (1NF)

关系模式: $SDC(\underline{Sno}, Sname, Sdept, Sdean, \underline{Cno}, Grade)$
函数依赖:
$Sno \to Sname, Sno \to Sdept, Sdept \to Sdean, (Sno, Cno) \to Grade$
结论:

- $SDC \in 1NF$
- (Sno, Cno)为SDC的候选键
- $(Sno, Cno) \xrightarrow{p} Sname$

1NF关系模式存在的问题

- 数据更新问题
  - 数据插入异常
  - 数据删除异常
  - 数据修改繁琐
- 数据冗余

1NF关系模式存在的问题(续)

问题1 (数据插入异常): 该插入的数据无法插入

Example (数据插入异常)

- 操作: 向SDC中插入一条学生元组，但该学生尚未选课
- 结果: 由于该学生尚未选课，该元组的Cno属性值为空，违反实体完整性约束，因此不能插入该元组

Sno	Sname	Sdept	Sdean	Cno	Grade
221101	Nick	Physics	Einstein	1002	92
221101	Nick	Physics	Einstein	1003	85
221101	Nick	Physics	Einstein	1006	88
231101	Elsa	CS	Turing	1006	90
231101	Elsa	CS	Turing	1003	95
231102	Eric	CS	Turing	1003	80
232101	Abby	Math	Gauss	1002	100
232102	John	Math	Gauss		

1NF关系模式存在的问题(续)

问题2 (数据删除异常): 不该删除的数据不得不删除

Example (数据删除异常)

- 操作: 某学生只选了一门课，而现在他连这门课也不选了，需要删除选课信息
- 结果: 由于Cno是SDC的主属性，取值不能为空，因此该学生整个元组都要被删除

Sno	Sname	Sdept	Sdean	Cno	Grade
221101	Nick	Physics	Einstein	1002	92
221101	Nick	Physics	Einstein	1003	85
221101	Nick	Physics	Einstein	1006	88
231101	Elsa	CS	Turing	1006	90
231101	Elsa	CS	Turing	1003	95
~~231102~~	~~Eric~~	~~CS~~	~~Turing~~	~~1003~~	~~80~~
232101	Abby	Math	Gauss	1002	100

1NF关系模式存在的问题(续)

问题3 (数据修改繁琐): 数据修改非常繁琐，容易出错

Example (数据修改繁琐)

- 操作: 某学生转系后，需要修改其所在系和系主任的信息
- 结果: 如果该学生选了n门课，则必须修改全部n个元组的Sdept和Sdean属性值，否则会导致数据不一致

Sno	Sname	Sdept	Sdean	Cno	Grade
221101	Nick	~~Physics~~ CS	~~Einstein~~ Turing	1002	92
221101	Nick	~~Physics~~ CS	~~Einstein~~ Turing	1003	85
221101	Nick	~~Physics~~ CS	~~Einstein~~ Turing	1006	88
231101	Elsa	CS	Turing	1006	90
231101	Elsa	CS	Turing	1003	95
231102	Eric	CS	Turing	1003	80
232101	Abby	Math	Gauss	1002	100

1NF关系模式存在的问题(续)

问题4 (数据冗余): 存在大量冗余数据

Example (数据冗余)

- 如果一名学生选了n门课，则他的姓名(Sname)、所在系(Sdept)和系主任(Sdean)的值要重复存储n次

Sno	Sname	Sdept	Sdean	Cno	Grade
221101	Nick	Physics	Einstein	1002	92
221101	Nick	Physics	Einstein	1003	85
221101	Nick	Physics	Einstein	1006	88
231101	Elsa	CS	Turing	1006	90
231101	Elsa	CS	Turing	1003	95
231102	Eric	CS	Turing	1003	80
232101	Abby	Math	Gauss	1002	100

1NF关系模式问题的解决方法

- 产生原因: 非主属性Sdept和Sdean部分函数依赖于候选键(Sno, Cno)
- 解决方法: 把SDC分解为下面两个关系模式[2]:
  - SD(Sno, Sname, Sdept, Sdean)
  - SC(Sno, Cno, Grade)

$SD = \Pi_{Sno,Sname,Sdept,Sdean}(SDC)$

Sno	Sname	Sdept	Sdean
221101	Nick	Physics	Einstein
231101	Elsa	CS	Turing
231102	Eric	CS	Turing
232101	Abby	Math	Gauss

$SC = \Pi_{Sno,Cno,Grade}(SDC)$

Sno	Cno	Grade
221101	1002	92
221101	1003	85
221101	1006	88
231101	1006	90
231101	1003	95
231102	1003	80
232101	1002	100

[2]SD代表Student-Department，SC代表Student-Course

1NF关系模式问题的解决方法(续)

解决了数据插入异常问题

Example

- 操作: 向SDC中插入一条学生元组，但该学生尚未选课
- 结果: 将该学生的信息插入到SD中，不存在插入异常

SD

Sno	Sname	Sdept	Sdean
221101	Nick	Physics	Einstein
231101	Elsa	CS	Turing
231102	Eric	CS	Turing
232101	Abby	Math	Gauss
232102	John	Math	Gauss

SC

Sno	Cno	Grade
221101	1002	92
221101	1003	85
221101	1006	88
231101	1006	90
231101	1003	95
231102	1003	80
232101	1002	100

1NF关系模式问题的解决方法(续)

解决了数据删除异常问题

Example

- 操作: 某学生只选了一门课，而现在他连这门课也不选了，需要删除选课信息
- 结果: 只需将该学生的选课信息从SC中删除，该学生在SD中的元组不受影响，因此不存在删除异常

SD

Sno	Sname	Sdept	Sdean
221101	Nick	Physics	Einstein
231101	Elsa	CS	Turing
231102	Eric	CS	Turing
232101	Abby	Math	Gauss

SC

Sno	Cno	Grade
221101	1002	92
221101	1003	85
221101	1006	88
231101	1006	90
231101	1003	95
~~231102~~	~~1003~~	~~80~~
232101	1002	100

1NF关系模式问题的解决方法(续)

简化了数据修改

Example

- 操作: 某学生转系后，需要修改其所在系和系主任的信息
- 结果: 只需在SD中将该学生所在系和系主任的信息修改1次，因此简化了修改操作

SD

Sno	Sname	Sdept	Sdean
221101	Nick	~~Physics~~ CS	~~Einstein~~ Turing
231101	Elsa	CS	Turing
231102	Eric	CS	Turing
232101	Abby	Math	Gauss

SC

Sno	Cno	Grade
221101	1002	92
221101	1003	85
221101	1006	88
231101	1006	90
231101	1003	95
231102	1003	80
232101	1002	100

1NF关系模式问题的解决方法(续)

减少了数据冗余

Example

- 无论一名学生选了几门课，该学生的姓名、所在系和系主任的信息只需在SD中存储1次，因此减少了数据冗余

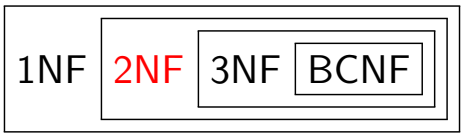
SD

Sno	Sname	Sdept	Sdean
221101	Nick	Physics	Einstein
231101	Elsa	CS	Turing
231102	Eric	CS	Turing
232101	Abby	Math	Gauss

SC

Sno	Cno	Grade
221101	1002	92
221101	1003	85
221101	1006	88
231101	1006	90
231101	1003	95
231102	1003	80
232101	1002	100

关系模式的范式



第二范式(2NF)

Definition (第二范式(2NF))

如果关系模式$R \in 1NF$，且R的每个非主属性都完全函数依赖于R的候选键，则称R为第二范式关系模式，记作$R \in 2NF$

Example (2NF)

1. $SDC(\underline{Sno}, Sname, Sdept, Sdean, \underline{Cno}, Grade) \notin 2NF$
   - 原因: $SDC \in 1NF$，但$(Sno, Cno) \xrightarrow{p} Sname$
2. $SD(\underline{Sno}, Sname, Sdept, Sdean) \in 2NF$
   - 原因: $SD \in 1NF$，且非主属性$Sname$、$Sdept$和$Sdean$均完全函数依赖于SD的候选键Sno
3. $SC(\underline{Sno, Cno}, Grade) \in 2NF$
   - 原因: $SC \in 1NF$，且非主属性$Grade$完全函数依赖于SC的候选键(Sno, Cno)

2NF关系模式存在的问题(续)

问题1 (数据插入异常): 该插入的数据无法插入

Example (数据插入异常)

- 操作: 插入某个系的信息，但该系刚成立，尚无学生
- 结果: 因为该系尚无学生，所以待插入元组的Sno属性值为空，违反了实体完整性约束，因此插入不进去

SD

Sno	Sname	Sdept	Sdean
221101	Nick	Physics	Einstein
231101	Elsa	CS	Turing
231102	Eric	CS	Turing
232101	Abby	Math	Gauss
		AI	Minsky

2NF关系模式存在的问题(续)

问题2 (数据删除异常): 不该删除的数据不得不删除

Example (数据删除异常)

- 操作: 某系学生全部毕业了，需要删除该系全体学生的信息
- 结果: 在删掉该系全部学生元组后，该系本身的信息也不复存在了

SD

Sno	Sname	Sdept	Sdean
221101	Nick	Physics	Einstein
~~231101~~	~~Elsa~~	~~CS~~	~~Turing~~
~~231102~~	~~Eric~~	~~CS~~	~~Turing~~
232101	Abby	Math	Gauss

2NF关系模式存在的问题(续)

问题3 (数据修改繁琐): 数据修改繁琐，容易出错

Example (数据修改繁琐)

- 操作: 某个系任命了新的系主任，需要修改该系的系主任信息
- 结果: 必须修改该系所有学生元组的Sdean属性值，否则会导致数据不一致

SD

Sno	Sname	Sdept	Sdean
221101	Nick	Physics	Einstein
231101	Elsa	CS	~~Turing~~ Shannon
231102	Eric	CS	~~Turing~~ Shannon
232101	Abby	Math	Gauss

2NF关系模式存在的问题(续)

问题4 (数据冗余): 存在大量冗余数据

Example (数据冗余)

- 现象: 如果某系有n名学生，那么该系系主任的信息要重复存储n次

Sno	Sname	Sdept	Sdean
221101	Nick	Physics	Einstein
231101	Elsa	CS	Turing
231102	Eric	CS	Turing
232101	Abby	Math	Gauss

2NF关系模式问题的解决方法

- 产生原因: 非主属性Sdean传递函数依赖于候选键Sno
- 解决方法: 把SD分解为下面两个关系模式:
  - S(Sno, Sname, Sdept)
  - D(Sdept, Sdean)

$S = \Pi_{Sno,Sname,Sdept}(SD)$

Sno	Sname	Sdept
221101	Nick	Physics
231101	Elsa	CS
231102	Eric	CS
232101	Abby	Math

$D = \Pi_{Sdept,Sdean}(SD)$

Sdept	Sdean
Physics	Einstein
CS	Turing
Math	Gauss

2NF关系模式问题的解决方法(续)

解决了数据插入异常问题

Example

- 操作: 插入某个系的信息，但该系刚成立，尚无学生
- 结果: 将该系的信息插入到D中，不存在插入异常

S

Sno	Sname	Sdept
221101	Nick	Physics
231101	Elsa	CS
231102	Eric	CS
232101	Abby	Math

D

Sdept	Sdean
Physics	Einstein
CS	Turing
Math	Gauss
AI	Minsky

2NF关系模式问题的解决方法(续)

解决了数据删除异常问题

Example

- 操作: 某系学生全部毕业了，需要删除该系全体学生的信息
- 结果: 只需将该系全体学生的信息从S中删除，该系在D中的信息不受影响，因此不存在删除异常

S

Sno	Sname	Sdept
221101	Nick	Physics
~~231101~~	~~Elsa~~	~~CS~~
~~231102~~	~~Eric~~	~~CS~~
232101	Abby	Math

D

Sdept	Sdean
Physics	Einstein
CS	Turing
Math	Gauss

2NF关系模式问题的解决方法(续)

简化了数据修改

Example

- 操作: 某个系任命了新的系主任，需要修改该系的系主任信息
- 结果: 只需修改D中该系的Sdean属性值，因此简化了数据修改

S

Sno	Sname	Sdept
221101	Nick	Physics
231101	Elsa	CS
231102	Eric	CS
232101	Abby	Math

D

Sdept	Sdean
Physics	Einstein
CS	~~Turing~~ Shannon
Math	Gauss

2NF关系模式问题的解决方法(续)

减少了数据冗余

Example

- 无论一个系有多少个学生，由于他们的系主任都相同，该系系主任的信息只需在D中存储1次

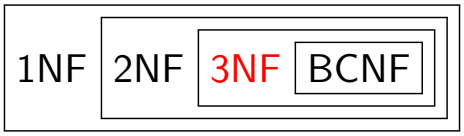
S

Sno	Sname	Sdept
221101	Nick	Physics
231101	Elsa	CS
231102	Eric	CS
232101	Abby	Math

D

Sdept	Sdean
Physics	Einstein
CS	Turing
Math	Gauss

关系模式的范式



第三范式(3NF)

Definition (第三范式(3NF))

如果关系模式$R \in 2NF$，且R的每个非主属性都不传递函数依赖于R的候选键，则称R为第三范式关系模式，记作$R \in 3NF$

Example (2NF)

1. $SD(\underline{Sno}, Sname, Sdept, Sdean) \notin 3NF$
   - 原因: $SD \in 2NF$，但非主属性$Sdean$传递函数依赖于SD的候选键Sno
2. $S(\underline{Sno}, Sname, Sdept) \in 3NF$
   - 原因: $S \in 2NF$，且非主属性$Sname$和$Sdept$均直接函数依赖于S的候选键Sno
3. $D(\underline{Sdept}, Sdean) \in 3NF$
   - 原因: $D \in 2NF$，且非主属性$Sdean$直接函数依赖于D的候选键$Sdept$

3NF关系模式存在的问题

将一个2NF关系模式分解为多个3NF关系模式，还是并不能完全消除数据更新异常和数据冗余问题

Example

- 考虑关系模式STC(Sno, Tno, Cno)，其中Sno为学号，Tno为教师号，Cno为课程号
- 该关系模式具有如下函数依赖
  - $(Sno, Cno) \rightarrow Tno$
  - $Tno \rightarrow Cno$ (假设一名教师只教一门课)
- 我们有如下结论
  - (Sno, Cno)和(Sno, Tno)都是STC的候选键
  - $STC \in 3NF$，因为不存在非主属性对候选键的传递函数依赖(STC中根本没有非主属性)

STC

Sno	Tno	Cno
S1	T1	C1
S2	T1	C1
S3	T2	C2
S4	T2	C2
S5	T3	C2

3NF关系模式存在的问题(续)

问题1 (数据插入异常): 该插入的数据无法插入

Example (数据插入异常)

- 操作: 某教师开了一门课，需要将该教师的开课信息插入数据库，但此时尚无学生选这门课
- 结果: 因为尚无学生选这门课，所以待插入元组的Sno属性值为空，违反了实体完整性约束，因此无法插入

STC

Sno	Tno	Cno
S1	T1	C1
S2	T1	C1
S3	T2	C2
S4	T2	C2
S5	T3	C2
	T4	C3

3NF关系模式存在的问题(续)

问题2 (数据删除异常): 不该删除的数据不得不删除

Example (数据删除异常)

- 操作: 选修过某门课的学生全部毕业了，需要从数据库中删除这些学生的信息
- 结果: 从STC中删掉这些学生元组后，这门课的信息也不复存在了

STC

Sno	Tno	Cno
S1	T1	C1
S2	T1	C1
~~S3~~	~~T2~~	~~C2~~
~~S4~~	~~T2~~	~~C2~~
~~S5~~	~~T3~~	~~C2~~

3NF关系模式存在的问题(续)

问题3 (数据修改繁琐): 数据修改繁琐，容易出错

Example (数据修改繁琐)

- 操作: 某教师开设的课程修改了课号，需要在数据库中修改该课程的课号
- 结果: 必须修改选修了该课程的所有学生元组的Cno属性值

STC

Sno	Tno	Cno
S1	T1	~~C1~~ C4
S2	T1	~~C1~~ C4
S3	T2	C2
S4	T2	C2
S5	T3	C2

3NF关系模式存在的问题(续)

问题4 (数据冗余): 存在大量冗余数据

Example (数据冗余)

- 现象: n名学生选了某教师的课程，尽管规定一名教师只能教一门课程，但该课程的信息还是要重复存储n次

STC

Sno	Tno	Cno
S1	T1	C1
S2	T1	C1
S3	T2	C2
S4	T2	C2
S5	T3	C2

3NF关系模式问题的解决方法

- 产生原因: 主属性Cno部分函数依赖于候选键(Sno, Tno)
- 解决方法: 把STC分解为下面两个关系模式:
  - TC(Tno, Cno)
  - ST(Sno, Tno)

$TC = \Pi_{Tno,Cno}(STC)$

Tno	Cno
T1	C1
T2	C2
T3	C2

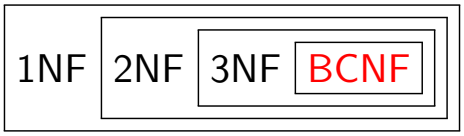
$ST = \Pi_{Sno,Tno}(STC)$

Sno	Tno
S1	T1
S2	T1
S3	T2
S4	T2
S5	T3

Example

该模式分解解决了STC存在的问题! 请同学们自己验证

关系模式的范式



Boyce-Codd范式(BCNF)

Definition (BC范式(BCNF))

1. 若关系模式$R(U,F) \in 1NF$，且F的最小覆盖中每个函数依赖$X \to Y$的左部X都是R的候选键，则称R为BC范式关系模式，记作$R \in BCNF$
2. 若关系模式$R(U,F) \in 3NF$，且R的任意主属性都直接完全函数依赖于R的候选键，则称R为BC范式关系模式，记作$R \in BCNF$

Example (BCNF)

1. $STC(Sno, Tno, Cno) \notin BCNF$
   - 原因: $STC \in 3NF$，但$Tno \to Cno$的左部不是候选键
2. $TC(Tno, Cno) \in BCNF$
   - 原因: $TC \in 3NF$，且$Tno \to Cno$的左部是候选键
3. $ST(Sno, Tno) \in BCNF$
   - 原因: $ST \in 3NF$，候选键是(Sno, Tno)

范式之间的关系

1NF
↓ 消除非主属性对候选键的部分函数依赖
2NF
↓ 消除非主属性对候选键的传递函数依赖
3NF
↓ 消除主属性对候选键的部分函数依赖和传递函数依赖
BCNF
↑ 消除决定属性集不是候选键的函数依赖
1NF

在函数依赖范畴内，BC范式已经达到了最高的范式级别

设计规范的关系模式

核心思想：遵循"一事一地"的模式设计原则，即一个关系模式只对应一个实体型或一个联系型

Example (设计规范的关系模式)

关系模式SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)中同时包含学生实体、系实体的信息，以及学生和课程之间联系的信息

SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)
↓ 将学生与课程之间联系的信息从SDC中分离出去
SD(Sno, Sname, Sdept, Sdean)
SC(Sno, Cno, Grade)
↓ 将系实体的信息从SD中分离出去
S(Sno, Sname, Sdept)
D(Sdept, Sdean)
SC(Sno, Cno, Grade)

误区：关系模式的规范化程度永远是越高越好

- 规范化程度越高，查询时需要进行的连接操作越多，查询效率越低

5.4 关系模式分解

Decomposition of Relational Schemas

关系模式分解

关系模式规范化是通过关系模式分解实现的

Definition (关系模式分解)

设$R(U)$是属性集U上的关系模式，$\rho = \{R_1(U_1), R_2(U_2), \ldots, R_n(U_n)\}$是$R$的一个分解(decomposition)，如果$U = U_1 \cup U_2 \cup \cdots \cup U_n$，且对任意$i \neq j$，有$U_i \not\subseteq U_j$ (无冗余)

Example (关系模式分解)

$\rho = \{S(Sno, Sname, Sdept), D(Sdept, Sdean), SC(Sno, Cno, Grade)\}$是对关系模式$SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)$的一个分解

- 关系模式分解的方法是不唯一的
- 在各种分解方法中，只有能够保证分解后的关系模式与原关系模式等价的分解方法才有意义

关系模式分解(续)

关系模式SDN(Sno, Sdept, Sdean)有多种分解方法

Example (分解方法1)

将SDN分解为S(Sno), D(Sdept), N(Sdean)

- $S, D, N \in BCNF$ (规范)
- $SDN \neq S \bowtie D \bowtie N$ (连接有损)

SDN

Sno	Sdept	Sdean
221101	Physics	Einstein
231101	CS	Turing
231102	CS	Turing
232101	Math	Gauss
241101	AI	Turing

$\Rightarrow$

S

Sno
221101
231101
231102
232101
241101

D

Sdept
Physics
CS
Math
AI

N

Sdean
Einstein
Turing
Gauss

关系模式分解(续)

Example (分解方法2)

将SDN分解为SN(Sno, Sdean), DN(Sdept, Sdean)

- $SN, DN \in BCNF$ (规范)
- $SDN \neq SN \bowtie DN$ (连接有损)

SDN

Sno	Sdept	Sdean
221101	Physics	Einstein
231101	CS	Turing
231102	CS	Turing
232101	Math	Gauss
241101	AI	Turing

SN

Sno	Sdean
221101	Einstein
231101	Turing
231102	Turing
232101	Gauss
241101	Turing

DN

Sdept	Sdean
Physics	Einstein
CS	Turing
Math	Gauss
AI	Turing

关系模式分解(续)

Example (分解方法3)

将SDN分解为SD(Sno, Sdept), SN(Sno, Sdean)

- $SD, SN \in BCNF$ (规范)
- $SDN = SD \bowtie SN$ (连接无损)
- $Sno \rightarrow Sdept$保留下来，但$Sdept \rightarrow Sdean$不存在了(未保持函数依赖)

SDN

Sno	Sdept	Sdean
221101	Physics	Einstein
231101	CS	Turing
231102	CS	Turing
232101	Math	Gauss
241101	AI	Turing

SD

Sno	Sdept
221101	Physics
231101	CS
231102	CS
232101	Math
241101	AI

SN

Sno	Sdean
221101	Einstein
231101	Turing
231102	Turing
232101	Gauss
241101	Turing

关系模式分解(续)

Example (分解方法4)

将SDN分解为SD(Sno, Sdept), DN(Sdept, Sdean)

- $SD, DN \in BCNF$ (规范)
- $SDN = SD \bowtie DN$ (连接无损)
- $Sno \rightarrow Sdept$和$Sdept \rightarrow Sdean$都保留下来了(保持函数依赖)

SDN

Sno	Sdept	Sdean
221101	Physics	Einstein
231101	CS	Turing
231102	CS	Turing
232101	Math	Gauss
241101	AI	Turing

SD

Sno	Sdept
221101	Physics
231101	CS
231102	CS
232101	Math
241101	AI

DN

Sdept	Sdean
Physics	Einstein
CS	Turing
Math	Gauss
AI	Turing

什么样的模式分解是好的模式分解?

关系模式分解准则1: 无损连接性(Lossless Join)

Definition (无损连接分解)

设$\rho = \{R_1(U_1), R_2(U_2), \ldots, R_n(U_n)\}$是对关系模式$R(U)$的一个分解，如果对于$R$的任意实例$r$，都有$r = \Pi_{U_1}(r) \bowtie \Pi_{U_2}(r) \bowtie \cdots \bowtie \Pi_{U_n}(r)$，则称$\rho$是$R$的无损连接分解

Example (无损连接分解)

{SD(Sno, Sdept), DN(Sdept, Sdean)}是对SDN的无损连接分解

SDN

Sno	Sdept	Sdean
221101	Physics	Einstein
231101	CS	Turing
231102	CS	Turing
232101	Math	Gauss
241101	AI	Turing

SD

Sno	Sdept
221101	Physics
231101	CS
231102	CS
232101	Math
241101	AI

DN

Sdept	Sdean
Physics	Einstein
CS	Turing
Math	Gauss
AI	Turing

无损连接分解可能存在的问题

对关系模式进行无损连接分解后，可能还存在数据更新异常、数据冗余等问题

Example (无损连接分解的问题)

- {SD(Sno, Sdept), SN(Sno, Sdean)}是对SDN的无损连接分解
- 操作: 221101号学生从物理系(Physics)转到计算机系(CS)
- 结果: 既要修改该学生在ST中的Sdept属性值，也要修改该学生在SN中的Sdean属性值，否则会破坏数据一致性
- 原因: SDN上原有的函数依赖$Sdept \rightarrow Sdean$在分解后丢失了

SD

Sno	Sdept
221101	~~Physics~~ CS
231101	CS
231102	CS
232101	Math
241101	AI

SN

Sno	Sdean
221101	~~Einstein~~ Turing
231101	Turing
231102	Turing
232101	Gauss
241101	Turing

无损连接分解可能存在的问题(续)

Example (无损连接分解的问题)

- {SD(Sno, Sdept), DN(Sdept, Sdean)}是对SDN的无损连接分解
- 操作: 221101号学生从物理系(Physics)转到计算机系(CS)
- 结果: 只需修改该学生在ST中的Sdept属性值
- 原因: SDN上原有的函数依赖$Sdept \to Sdean$在DN中保留了

SD

Sno	Sdept
221101	~~Physics~~ CS
231101	CS
231102	CS
232101	Math
241101	AI

DN

Sdept	Sdean
Physics	Einstein
CS	Turing
Math	Gauss
AI	Turing

关系模式分解准则2: 函数依赖保持性

Definition (函数依赖集的投影)

设$R(U, F)$是关系模式，U是属性集，F是函数依赖集。对于$V \subseteq U$，F在V上的投影是F^+中满足$X \cup Y \subseteq V$的函数依赖$X \to Y$的集合

Example (函数依赖集的投影)

SDN上的函数依赖集$F = \{Sno \to Sdept, Sdept \to Sdean\}$

- F在(Sno, Sdept)上的投影是$\{Sno \to Sdept\}$
- F在(Sno, Sdean)上的投影是$\{Sno \to Sdean\}$ (并未直接出现在F中)
- F在(Sdept, Sdean)上的投影是$\{Sdept \to Sdean\}$

Definition (保持函数依赖分解)

设$R(U, F)$是一个关系模式，U是属性集，F是函数依赖集。
设$\rho = \{R_1(U_1), R_2(U_2), \ldots, R_n(U_n)\}$是对$R$的一个分解，$F_i$是$F$在$U_i$上的投影。如果$F_1 \cup F_2 \cup \cdots \cup F_n$覆盖$F$，则$\rho$是保持函数依赖的模式分解。

关系模式分解准则2: 函数依赖保持性(续)

Example (保持函数依赖的分解)

SDN上的函数依赖集为$F = \{Sno \to Sdept, Sdept \to Sdean\}$。
将SDN分解为SD(Sno, Sdept), SN(Sno, Sdean)

- F在SD(Sno, Sdept)上的投影为$F_{SD} = \{Sno \to Sdept\}$
- F在SN(Sno, Sdean)上的投影为$F_{SN} = \{Sno \to Sdean\}$
- $F_{SD} \cup F_{SN}$不能覆盖F (如何证明?)
- 该分解不是保持函数依赖的分解

Example (保持函数依赖的分解)

将SDN分解为SD(Sno, Sdept), DN(Sdept, Sdean)

- F在SD(Sno, Sdept)上的投影为$F_{SD} = \{Sno \to Sdept\}$
- F在DN(Sdept, Sdean)上的投影为$F_{DN} = \{Sdept \to Sdean\}$
- $F_{SD} \cup F_{SN} = F$，一定能覆盖F
- 该分解是保持函数依赖的分解

关系模式分解准则(续)

- 一个好的关系模式分解需要满足的特性
  - 无损连接性: 确保分解后数据不丢失
  - 函数依赖保持性: 减轻或解决异常情况
- 无损连接性和函数依赖保持性是两个相互独立的准则
  - 无损连接分解不一定是保持函数依赖的分解
  - 保持函数依赖的分解也不一定是无损连接分解
- 如何判断一个关系模式分解是否满足无损连接性?
- 如何判断一个关系模式分解是否满足函数依赖保持性?

无损连接性的判定

- 目的: 判定一个关系模式分解是否满足无损连接性
- 方法1 (一个关系模式被分解为两个关系模式时的判定方法)

Theorem

关系模式$R(U,F)$的分解$\rho=\{R_1(U_1),R_2(U_2)\}$满足无损连接性的充要条件是$F\models U_1\cap U_2\to U_1$或$F\models U_1\cap U_2\to U_2$

Example (无损连接性的判定1)

SDN上的函数依赖集为$F=\{Sno\to Sdept, Sdept\to Sdean\}$。
将SDN分解为SD(Sno, Sdept), SN(Sno, Sdean)

- $Sno\to(Sno, Sdept)$
- $Sno\to(Sno, Sdean)$
- 该分解是无损连接分解

无损连接性的判定(续)

Example (无损连接性的判定2)

将SDN分解为SD(Sno, Sdept), DN(Sdept, Sdean)

- $Sdept\twoheadrightarrow(Sno, Sdept)$
- $Sdept\to(Sdept, Sdean)$
- 该分解是无损连接分解

Example (无损连接性的判定3)

将SDN分解为SN(Sno, Sdean), DN(Sdept, Sdean)

- $Sdean\twoheadrightarrow(Sno, Sdean)$
- $Sdean\twoheadrightarrow(Sdept, Sdean)$
- 该分解不是无损连接分解

无损连接性的判定(续)

- 目的: 判定一个关系模式分解是否满足无损连接性
- 方法2 (一个关系模式被分解为3个以上关系模式时的判定方法)

Example

- 属性集$U = \{A, B, C, D, E\}$
- 函数依赖集$F = \{A \to C, B \to C, C \to D, DE \to C, CE \to A\}$
- 关系模式分解$\rho = \{(A, D), (A, B), (B, E), (C, D, E), (A, E)\}$

1. 初始化

	A	B	C	D	E
$R_1(A, D)$	**a**	$x_{1,2}$	$x_{1,3}$	**d**	$x_{1,5}$
$R_2(A, B)$	**a**	**b**	$x_{2,3}$	$x_{2,4}$	$x_{2,5}$
$R_3(B, E)$	$x_{3,1}$	**b**	$x_{3,3}$	$x_{3,4}$	**e**
$R_4(C, D, E)$	$x_{4,1}$	$x_{4,2}$	**c**	**d**	**e**
$R_5(A, E)$	**a**	$x_{5,2}$	$x_{5,3}$	$x_{5,4}$	**e**

无损连接性的判定(续)

Example

- 函数依赖集$F = \{A \to C, B \to C, C \to D, DE \to C, CE \to A\}$

2. 根据$A \to C$，可知$x_{1,3} = x_{2,3} = x_{5,3}$

	A	B	C	D	E
$R_1(A, D)$	**a**	$x_{1,2}$	$x_{1,3}$	**d**	$x_{1,5}$
$R_2(A, B)$	**a**	**b**	$x_{2,3} = x_{1,3}$	$x_{2,4}$	$x_{2,5}$
$R_3(B, E)$	$x_{3,1}$	**b**	$x_{3,3}$	$x_{3,4}$	**e**
$R_4(C, D, E)$	$x_{4,1}$	$x_{4,2}$	**c**	**d**	**e**
$R_5(A, E)$	**a**	$x_{5,2}$	$x_{5,3} = x_{1,3}$	$x_{5,4}$	**e**

3. 根据$B \to C$，可知$x_{1,3} = x_{3,3}$

	A	B	C	D	E
$R_1(A, D)$	**a**	$x_{1,2}$	$x_{1,3}$	**d**	$x_{1,5}$
$R_2(A, B)$	**a**	**b**	$x_{1,3}$	$x_{2,4}$	$x_{2,5}$
$R_3(B, E)$	$x_{3,1}$	**b**	$x_{3,3} = x_{1,3}$	$x_{3,4}$	**e**
$R_4(C, D, E)$	$x_{4,1}$	$x_{4,2}$	**c**	**d**	**e**
$R_5(A, E)$	**a**	$x_{5,2}$	$x_{1,3}$	$x_{5,4}$	**e**

无损连接性的判定(续)

Example

- 函数依赖集$F = \{A \to C, B \to C, C \to D, DE \to C, CE \to A\}$

4. 根据$C \to D$，可知$x_{2,4} = x_{3,4} = x_{5,4} = \mathbf{d}$

	A	B	C	D	E
$R_1(A, D)$	**a**	$x_{1,2}$	$x_{1,3}$	**d**	$x_{1,5}$
$R_2(A, B)$	**a**	**b**	$x_{1,3}$	$x_{2,4} = \mathbf{d}$	$x_{2,5}$
$R_3(B, E)$	$x_{3,1}$	**b**	$x_{1,3}$	$x_{3,4} = \mathbf{d}$	**e**
$R_4(C, D, E)$	$x_{4,1}$	$x_{4,2}$	**c**	**d**	**e**
$R_5(A, E)$	**a**	$x_{5,2}$	$x_{1,3}$	$x_{5,4} = \mathbf{d}$	**e**

5. 根据$DE \to C$，可知$x_{1,3} = \mathbf{c}$

	A	B	C	D	E
$R_1(A, D)$	**a**	$x_{1,2}$	$x_{1,3} = \mathbf{c}$	**d**	$x_{1,5}$
$R_2(A, B)$	**a**	**b**	$x_{1,3} = \mathbf{c}$	**d**	$x_{2,5}$
$R_3(B, E)$	$x_{3,1}$	**b**	$x_{1,3} = \mathbf{c}$	**d**	**e**
$R_4(C, D, E)$	$x_{4,1}$	$x_{4,2}$	**c**	**d**	**e**
$R_5(A, E)$	**a**	$x_{5,2}$	$x_{1,3} = \mathbf{c}$	**d**	**e**

无损连接性的判定(续)

Example

- 函数依赖集$F = \{A \to C, B \to C, C \to D, DE \to C, CE \to A\}$

6. 根据$CE \to A$，可知$x_{3,1} = x_{4,1} = \mathbf{a}$

	A	B	C	D	E
$R_1(A, D)$	**a**	$x_{1,2}$	**c**	**d**	$x_{1,5}$
$R_2(A, B)$	**a**	**b**	**c**	**d**	$x_{2,5}$
$R_3(B, E)$	$x_{3,1} = \mathbf{a}$	**b**	**c**	**d**	**e**
$R_4(C, D, E)$	$x_{4,1} = \mathbf{a}$	$x_{4,2}$	**c**	**d**	**e**
$R_5(A, E)$	**a**	$x_{5,2}$	**c**	**d**	**e**

7. 发现表中有一行是$(\mathbf{a}, \mathbf{b}, \mathbf{c}, \mathbf{d}, \mathbf{e})$，说明该分解是无损连接分解

	A	B	C	D	E
$R_1(A, D)$	**a**	$x_{1,2}$	**c**	**d**	$x_{1,5}$
$R_2(A, B)$	**a**	**b**	**c**	**d**	$x_{2,5}$
$R_3(B, E)$	**a**	**b**	**c**	**d**	**e**
$R_4(C, D, E)$	**a**	$x_{4,2}$	**c**	**d**	**e**
$R_5(A, E)$	**a**	$x_{5,2}$	**c**	**d**	**e**

无损连接性的判定(续)

为什么表中有一行是$(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d},\mathbf{e})$就能说明该分解是无损连接分解?

1. 显然，$R \subseteq \Pi_{A,D}(R) \bowtie \Pi_{A,B}(R) \bowtie \Pi_{B,E}(R) \bowtie \Pi_{C,D,E}(R) \bowtie \Pi_{A,E}(R) = R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \bowtie R_5$
2. 往证$R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \bowtie R_5 \subseteq R$
   - 对任意$(\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d},\mathbf{e}) \in R_1 \bowtie R_2 \bowtie R_3 \bowtie R_4 \bowtie R_5$，它是由以下元组连接而成的
     - $(\mathbf{a},\mathbf{d}) \in R_1$
     - $(\mathbf{a},\mathbf{b}) \in R_2$
     - $(\mathbf{b},\mathbf{e}) \in R_3$
     - $(\mathbf{c},\mathbf{d},\mathbf{e}) \in R_4$
     - $(\mathbf{a},\mathbf{e}) \in R_5$
   - 这些元组又是通过R中以下元组投影得到的($x_{i,j}$代表未知数)
     - $(\mathbf{a}, x_{1,2}, x_{1,3}, \mathbf{d}, x_{1,5}) \in R \xrightarrow{projection} (\mathbf{a},\mathbf{d}) \in R_1$
     - $(\mathbf{a}, \mathbf{b}, x_{2,3}, x_{2,4}, x_{2,5}) \in R \xrightarrow{projection} (\mathbf{a},\mathbf{b}) \in R_2$
     - $(x_{3,1}, \mathbf{b}, x_{3,3}, x_{3,4}, \mathbf{e}) \in R \xrightarrow{projection} (\mathbf{b},\mathbf{e}) \in R_3$
     - $(x_{4,1}, x_{4,2}, \mathbf{c}, \mathbf{d}, \mathbf{e}) \in R \xrightarrow{projection} (\mathbf{c},\mathbf{d},\mathbf{e}) \in R_4$
     - $(\mathbf{a}, x_{5,2}, x_{5,3}, x_{5,4}, \mathbf{e}) \in R \xrightarrow{projection} (\mathbf{a},\mathbf{e}) \in R_5$
   - 根据例子，我们知道$(x_{3,1}, \mathbf{b}, x_{3,3}, x_{3,4}, \mathbf{e}) = (\mathbf{a},\mathbf{b},\mathbf{c},\mathbf{d},\mathbf{e}) \in R$

函数依赖保持性的判定

- 目的: 判定一个关系模式分解是否满足函数依赖保持性
- 方法1: 根据函数依赖保持性的定义进行判断
- 方法2: 对于函数依赖集F中每个函数依赖$X \to Y$，使用下面的算法判断是否$F_1 \cup F_2 \cup \cdots \cup F_n \models X \to Y$

函数依赖保持性判定

输入: 关系模式R的函数依赖集F、函数依赖$X \to Y \in F$、R的分解$\rho = \{R_1(U_1), R_2(U_2), \ldots, R_n(U_n)\}$
输出: $X \to Y$是否被$F_1 \cup F_2 \cup \cdots \cup F_n$逻辑蕴含

```
1: X+ ← X
2: repeat
3:     for i = 1 to n do
4:         X+ ← X+ ∪ ((X+ ∩ Ui)+F ∩ Ui)
5: until X+不再变化
6: if Y ⊆ X+ then
7:     return true
8: else
9:     return false
```

函数依赖保持性的判定(续)

函数依赖保持性判定

1: $X^+ \leftarrow X$
2: **repeat**
3: **for** $i = 1$ **to** n **do**
4: $X^+ \leftarrow X^+ \cup ((X^+ \cap U_i)_F^+ \cap U_i)$
5: **until** X^+不再变化
6: **if** $Y \subseteq X^+$ **then**
7: **return** **true**
8: **else**
9: **return** **false**

Example (函数依赖保持性判定)

SDN上的函数依赖集为$F = \{Sno \rightarrow Sdept, Sdept \rightarrow Sdean\}$。
将SDN分解为SD(Sno, Sdept), SN(Sno, Sdean)

1. 显然，SD保持了$Sno \rightarrow Sdept$
2. 因为$\{Sdean\} \not\subseteq Sdept^+$，所以$Sdept \rightarrow Sdean$未被保持

关系模式的分解方法

如何对一个设计得不够规范的关系模式进行分解?

- 人工分解: 由数据库设计者根据关系模式上的函数依赖对关系模式进行人工分解
  - 优点: 对关系模式的语义具有深入理解，不会"过度分解"
  - 缺点: 只能对属性较少的关系模式进行分解
- 关系模式分解算法: 使用算法对关系模式进行自动分解
  - 优点: 能够对具有很多属性的关系模式进行自动分解
  - 缺点: 缺少对关系模式语义的理解，可能会"过度分解"

3NF关系模式分解算法

- 功能: 将关系模式$R(U,F)$分解为一组3NF关系模式，且该分解既保持函数依赖，又满足无损连接性

3NF关系模式分解算法

1: $G \leftarrow F$的最小覆盖
2: **if** 存在$X \to Y \in G$使得$X \cup Y = U$ **then**
3: **return** $\{R\}$
4: $\rho \leftarrow \{R_K(K)\}$ (K是R的候选键)
5: $U_0 \leftarrow$ 不出现在G中任何函数依赖中的属性的集合
6: **if** $U_0 \neq \emptyset$ **then**
7: $\rho \leftarrow \rho \cup \{R_0(U_0)\}$
8: $i \leftarrow 1$
9: **for all** $X \to Y \in G$ **do**
10: $\rho \leftarrow \rho \cup \{R_i(X \cup Y)\}$
11: $i \leftarrow i + 1$
12: 从ρ中去除被其他关系模式包含的关系模式
13: **return** ρ

3NF关系模式分解算法(续)

Example (3NF关系模式分解算法)

- 关系模式SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)
- 函数依赖集$F = \{Sno \to Sname, Sno \to Sdept, Sdept \to Sdean, (Sno, Cno) \to Grade\}$

F的最小覆盖	分解后的关系模式	是否冗余
	~~$R_K(Sno, Cno)$~~	是
$Sno \to (Sname, Sdept)$	$R_1(Sno, Sname, Sdept)$	否
$Sdept \to Sdean$	$R_2(Sdept, Sdean)$	否
$(Sno, Cno) \to Grade$	$R_3(Sno, Cno, Grade)$	否

分解后的关系模式	范式	F的投影
$R_1(Sno, Sname, Sdept)$	BCNF	$F_1 = \{Sno \to (Sname, Sdept)\}$
$R_2(Sdept, Sdean)$	BCNF	$F_2 = \{Sdept \to Sdean\}$
$R_3(Sno, Cno, Grade)$	BCNF	$F_3 = \{(Sno, Cno) \to Grade\}$

$F_1 \cup F_2 \cup F_3$覆盖F，所以保持函数依赖

BCNF关系模式分解算法

- 功能: 将关系模式$R(U,F)$分解为一组无损连接BCNF关系模式

BCNF关系模式分解算法

输入: 关系模式$R(U,F)$
输出: R的无损连接BCNF关系模式分解ρ

1: $\rho \leftarrow \{R\}$
2: 计算F^+ {指数时间复杂度}
3: **repeat**
4: **if** ρ中存在关系模式$R_i(U_i)$不属于BCNF **then**
5: 从F_i^+中找出非平凡函数依赖$X \rightarrow Y$，满足$X \cap Y = \emptyset$且X不是R_i的候选键
6: $\rho \leftarrow (\rho - R_i(U_i)) \cup \{R_{i1}(XY), R_{i2}(U_i - Y)\}$
7: **until** ρ不再变化
8: **return** ρ

总结

1. ER模型转换为关系数据库模式
   - 实体相关的概念→关系模式: 实体型、复合属性、多值属性、弱实体型的转换
   - 联系相关的概念→关系模式: M:N联系型、N:1联系型、二元自联系型的转换
2. 函数依赖
   - 函数依赖的类型: 完全函数依赖、部分函数依赖、传递函数依赖
   - Armstrong公理系统: 自反律、增广律、传递律
   - 属性集的闭包: 概念、计算算法、如何用于推理
   - 等价函数依赖集: 概念、证明方法
   - 函数依赖集的最小覆盖: 概念、计算算法
3. 关系模式的范式
   - 范式的种类: $4NF \subseteq BCNF \subseteq 3NF \subseteq 2NF \subseteq 1NF$
   - 不规范关系模式存在的问题、成因及解决方法
4. 关系模式分解
   - 关系模式分解的准则: 无损连接性、函数依赖保持性
   - 无损连接性的判定
   - 函数依赖保持性的判定
   - 关系模式分解算法

习题 I

1. 在将ER模型转换为关系数据库模式时，为什么要将多值属性转换为关系?
2. 在将N:1二元联系型转换为关系模式时，为什么可以将基数1一方实体型的主键并入基数N一方的实体型关系的属性集中?
3. 在将1:1二元联系型转换为关系模式时，什么情况下可将任意一方实体型的主键并入另一方的实体型关系的属性集中?
4. 为什么标识联系型不用转换成关系模式?
5. 以下说法是否正确? 在某关系实例中，对任意2条元组t_1和t_2，如果由$t_1[X] = t_2[X]$可以推出$t_1[Y] = t_2[Y]$，则$X \to Y$
6. 使用函数依赖的概念定义候选键
7. 证明Armstrong公理的3条导出规则成立
8. 关系模式SDC(Sno, Sname, Sdept, Sdean, Cno, Grade)上的函数依赖集为$F = \{Sno \to Sname, Sno \to Sdept, Sdept \to Sdean, (Sno, Cno) \to Grade\}$，用多种方法证明$F \models (Sno, Cno) \to (Sdean, Grade)$