数据库系统概论
高级篇
An Introduction to Database System

数据库系统概论 基础篇

基础篇讲解数据库系统的基本概念、基础知识和基本技术

教材的第一篇 基础篇

- 第一章 绪论 初步认识数据库系统

  数据库的产生和发展、数据库的特点、3级模式架构、数据库系统组成

- 第二章 关系数据库

  讲解关系数据库的数据结构、完整性约束条件、关系代数

- 第三章 关系数据库标准语言SQL

  如何用SQL定义、存取数据库中的数据

- 第四章 数据库安全性

  如何保护数据库以防止不合法使用所造成的数据泄露、更改或破坏 。

- 第五章 数据库完整性

  如何保证数据库中数据的准确性和有效性

数据库系统概论（高级篇）

讲授内容包括：

教材的第二篇 设计与应用开发篇

讲解应用系统开发过程中如何基于某个DBMS设计数据库

- 第6章 关系数据理论
- 第7章 数据库设计

教材的第三篇 系统篇

讨论DBMS中查询处理，事务管理，数据库恢复和并发控制等基本概念和基本技术

- 第9章 关系查询处理和查询优化(简单介绍)
- 第10章 数据库恢复技术
- 第11章 并发控制

第六章 关系数据理论

第六章 关系数据理论

6.1 问题的提出—为什么要学习关系数据理论
6.2 规范化
6.3 数据依赖的公理系统（简单介绍）
6.4 模式的分解（简单介绍）
6.5 小结

第六章 关系数据理论

6.1 问题的提出—为什么要学习关系数据理论

6.2 规范化

6.3 数据依赖的公理系统（简单介绍）

6.4 模式的分解（简单介绍）

6.5 小结

6.1 问题的提出

- 关系数据库的基本概念
- 关系模型
- 关系数据库的标准语言
- 关系数据库逻辑设计

  针对一个具体问题，应如何构造一个适合于它的数据模式，即应该构造几个关系，每个关系由哪些属性组成等。

问题——什么是一个好的数据库逻辑设计

一个例子

- 学校开发一个学校教务的数据库，涉及的对象有：
  学生的学号（Sno）、所在系（Sdept）、系主任姓名（Mname）、
  课程号（Cno）和成绩（Grade）。
- 语义：
  1. 一个系有若干学生， 但一个学生只属于一个系；
  2. 一个系只有一名主任；
  3. 一个学生可以选修多门课程， 每门课程有若干学生选修；
  4. 每个学生所学的每门课程都有一个成绩。
- 存储并管理这些信息
- 设计了一个关系模式
  STUDENT（Sno ,Sdept, Mname,Cno,Grade）

一个例子

表6.1 Student表

Sno	Sdept	Mname	Cno	Grade
S1	计算机系	张明	C1	95
S2	计算机系	张明	C1	90
S3	计算机系	张明	C1	88
S4	计算机系	张明	C1	70
S5	计算机系	张明	C1	78
⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮	⋮ ⋮

关系模式存在的问题

关系模式STUDENT（Sno ，Sdept， Mname,Cno,Grade）中存在的问题：

1. 数据冗余度太大，浪费存储空间

如：系主任的姓名重复出现，重复次数与该系所有学生的所有课程成绩出现次数相同。

2. 更新异常（Update Anomalies）

数据冗余 ，更新数据时，维护数据完整性代价大

如果某系更换系主任，系统必须修改与该系学生有关的每一个元组。

Sno	Sdept	Mname	Cno	Grade
S1	计算机系	张明	C1	95
S2	计算机系	张明	C1	90
S3	计算机系	张明	C1	88
S4	计算机系	张明	C1	70
S5	计算机系	张明	C1	78

关系模式存在的问题

3. 插入异常（Insertion Anomalies），该插入的数据插不进去

如果新成立一个软件工程系，还没有招生，我们就无法把这个系及其系主任的信息存入数据库。

4. 删除异常（Deletion Anomalies），不该删除的数据也删去了

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，把这个系及其系主任的信息也丢掉了。

Sno	Sdept	Mname	Cno	Grade
S1	计算机系	张明	C1	95
S2	计算机系	张明	C1	90
S3	计算机系	张明	C1	88
S4	计算机系	张明	C1	70
S5	计算机系	张明	C1	78

NULL	软件工程系	严广	null	null

STUDENT（Sno ，Sdept，Mname，Cno，Grade）
不是一个好的关系模式。

关系模式存在的问题

- 什么是一个好的模式。

  好的模式不会发生插入异常、删除异常、更新异常、数据冗余应尽可能少。

- 问题的原因

  由于模式中的某些数据依赖引起的。

解决方法：把这个单一模式分成3个关系模式：

S（Sno，Sdept，Sno → Sdept）；

SC（Sno，Cno，Grade，（Sno，Cno） → Grade）；

DEPT（Sdept，Mname，Sdept→ Mname）

这3个模式不会发生插入异常、删除异常毛病；数据冗余得到控制。

再把观察、经验上升为理论：

用规范化理论改造关系模式，消除其中不合适的数据依赖。

6.1 问题的提出

1. 问题——什么是一个好的数据库逻辑设计
2. 什么是数据依赖
3. 关系模式的简化表示

2. 什么是数据依赖：一个例子

- STUDENT（Sno ，Sdept， Mname，Cno，Grade）
- 该关系模式的属性集合记为U：

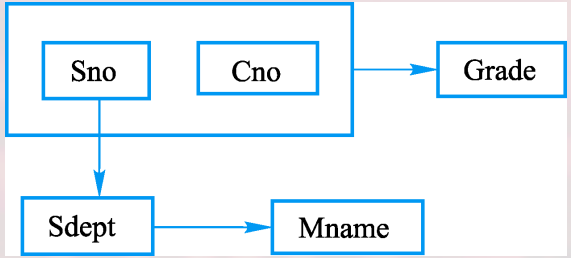
  U ＝ { Sno， Sdept， Mname， Cno， Grade }

- 数学中的函数*y=f(x)*，自变量*x*确定之后，相应的函数值*y*也就唯一地确定了。
- Sdept=f(Sno), Sno函数确定Sdept，记为Sno→Sdept,
- Mname=f(Sdept), Sdept函数确定Mname，记为Sdept→Mname
- Grade=f((Sno, Cno)) , (Sno, Cno)函数确定Grade，记为(Sno, Cno)→Grade

- 属性组U上的函数依赖集合，记为F：

  F ＝ { Sno→Sdept， Sdept→Mname， (Sno， Cno)→Grade}

用一个图表示函数依赖：



什么是数据依赖

1. 完整性约束的一种表现形式

- 定义属性值间的相互关连（主要体现于值的相等与否）
  即通过属性间值的相等与否来描述
- 它是数据库模式设计的关键

什么是数据依赖（续）

2. 数据依赖

- 是通过一个关系中属性间值的相等与否体现出来的数据间的相互关系
- 是现实世界属性间相互联系的抽象
- 是数据内在的性质
- 是语义的体现

什么是数据依赖

3. 数据依赖的主要类型

- 函数依赖(Functional Dependency，简记为FD)
- 多值依赖(Multivalued Dependency，简记为MVD)
- 连接依赖
- … …

4. 数据依赖对关系模式的影响

- 不合适的数据依赖，造成插入异常、删除异常、更新异常和数据冗余问题

6.1 问题的提出

1.问题——什么是一个好的数据库逻辑设计

2.什么是数据依赖

3.关系模式的简化表示

3. 关系模式的简化表示

1. 关系模式的形式化定义 （2.1.2关系模式 P.42讲解过）

R(U, D, DOM, F)

- R：关系名，是符号化的元组语义
- U：该关系的属性集合
- D：属性组U中属性所来自的域
- DOM：属性向域的映象集合
- F：属性间数据的依赖关系集合

2. 关系模式的简化表示

R<U, F>

将关系模式简化为一个三元组，影响数据库模式设计的主要是 U 和 F.

当且仅当U上的一个关系 r 满足F时，r 称为关系模式R(U, F)的一个关系。

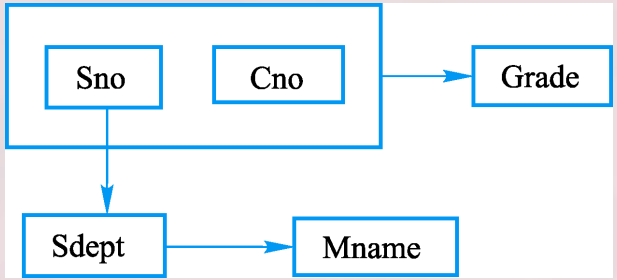
再看例子

关系模式 STUDENT<U,F>

U = { Sno,Sdept,Mname,Cno,Grade }

F = { Sno→Sdept, Sdept→Mname, (Sno, Cno)→Grade}

STUDENT（Sno,Sdept,Mname,Cno,Grade, Sno→Sdept, Sdept→Mname, (Sno, Cno)→Grade）



关系模式 STUDENT<U,F>存在诸多问题。

如何解决关系模式中存在的问题？

规范化理论—找出关系模式中不合适的数据依赖，消除它们，可以在不同程度上解决插入异常、删除异常、更新异常和数据冗余问题。

第六章 关系数据理论

6.1 问题的提出—为什么要学习关系数据理论

6.2 规范化

6.3 数据依赖的公理系统（简单介绍）

6.4 模式的分解（简单介绍）

6.5 小结

6.2 规范化—关系的规范化理论

6.2 规范化—关系的规范化理论

6.2.1 函数依赖

1. 函数依赖
2. 平凡函数依赖与非平凡函数依赖
3. 完全函数依赖与部分函数依赖
4. 传递函数依赖

1. 函数依赖

定义6.1

设R(U)是一个属性集U上的关系模式，X和Y是U的子集。若对于R(U)的任意一个可能的关系r，r中不可能存在两个元组在X上的属性值相等， 而在Y上的属性值不等则称“**X函数确定Y**”或“**Y函数依赖于X**”，记作**X→Y**。

X称为这个函数依赖的决定属性组，也称为**决定因素**(Determinant)。

例： S(Sno, Sname, Ssex, Sage, Sdept)

F＝ {Sno→Sname，Sno→Ssex，Sno→Sage，Sno→Sdept}

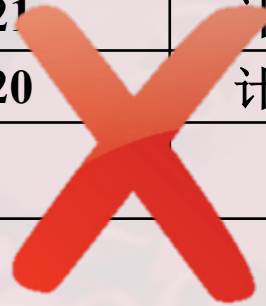
Ssex $\not\rightarrow$ Sage, Ssex $\not\rightarrow$ Sdept

若Y不函数依赖于X, 则记为X $\not\rightarrow$ Y。

1. 函数依赖（续）

违背了Sno → Sname

Sno	Sname	Ssex	Sage	Sdept
S1	张三	男	20	计算机系
S1	李四	女	21	自动化系
S3	王五	男	20	计算机系
S4	赵六	男	21	计算机系
S5	田七	男	20	计算机系
⋮	⋮	⋮	⋮	⋮



如何确定函数依赖（续）

由下面的关系表，能否得出**Sname → Sno ?**

Sno	Sname	Ssex	Sage	Sdept
S1	张三	男	20	计算机系
S2	李四	女	21	自动化系
S3	王五	男	20	计算机系
S4	赵六	男	21	计算机系
S5	田七	男	20	计算机系
⋮	⋮	⋮	⋮	⋮

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

如何确定函数依赖

- 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。**
  - 如Sname →Sno函数依赖只有在“学生不允许有重名”的条件下成立。
- 数据库设计者可以对现实世界作强制的规定。
  - 例如设计者**可以强行规定不允许学生有重名**，因而使函数依赖 Sname →Sno，Sname →Ssex， Sname →Sage，Sname→Sdept 成立。
- 函数依赖是指**关系模式R在任何时刻的关系实例均要满足的约束条件。**
  - 不是指某个或某些关系实例 r 满足的约束条件，而是指R的所有关系实例 r 均要满足的约束条件。

2. 平凡函数依赖与非平凡函数依赖

- X→Y，Y⊈X，则称X→Y是非平凡的函数依赖。
- X→Y，但Y⊆X，则称X→Y是平凡的函数依赖。

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

对于任一关系模式，平凡函数依赖都是必然成立的，它不反映新的语义，因此若不特别声明，**我们总是讨论非平凡函数依赖。**

3. 完全函数依赖与部分函数依赖

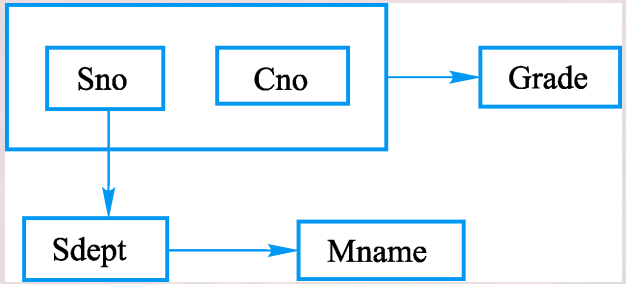
定义6.2

在关系模式R(U)中，如果X→Y，并且对于X的任何一个真子集X'，都有X'↛Y，则称Y完全函数依赖于X，记作$X \xrightarrow{F} Y$。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，记作$X \xrightarrow{P} Y$。

[例] 在关系STUDENT（Sno ,Sdept, Mname,Cno,Grade）中，

(Sno, Cno) $\xrightarrow{F}$ Grade 是完全函数依赖

(Sno, Cno) $\xrightarrow{P}$ Sdept 是部分函数依赖，因为Sno → Sdept，



4. 传递函数依赖

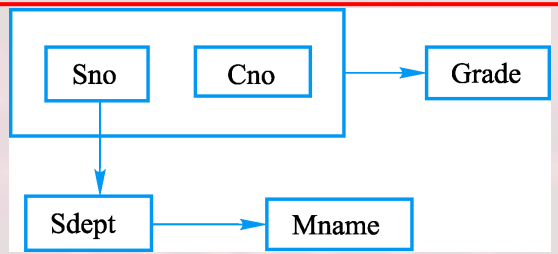
定义6.3

在R（U）中，如果X→Y，（Y⊈X），Y↛X，Y→Z，则称Z对X传递函数依赖（transitive functional dependency）。记为：$X \xrightarrow{传递} Z$。

注意: 如果Y→X, 即X←→Y，则Z直接依赖于X。

[例] 在关系STUDENT（Sno ,Sdept, Mname,Cno,Grade）中，

Sno → Sdept，Sdept → Mname，$Sno \xrightarrow{传递} Mname$



思考与讨论

- 有关系R（C,D,H,S），
- 从 r1、r2 是否可以说 C→DH 成立？

关系 r 1

C	D	H	S
c1	d1	h1	s1
c1	d1	h1	s1
c1	d1	h1	s2
c2	d2	h2	s3

关系 r 2

C	D	H	S
c1	d1	h1	s1
c1	d1	h2	s1
c1	d1	h1	s2
c2	d2	h2	s3

一般地，从 关系实例，我们能确定某个函数依赖成立吗？

6.2.1 函数依赖回顾

WHAT 什么是函数依赖

完全函数依赖

部分函数依赖

传递函数依赖

HOW 如何确定函数依赖

WHY 为什么要学习函数依赖

6.2 规范化—关系的规范化理论

6.2.1 函数依赖

6.2.2 码

6.2.3 范式

6.2.4 第二范式（2NF）

6.2.5 第三范式（3NF）

6.2.6 BC范式（BCNF）

*6.2.7 多值依赖

*6.2.8 第四范式（4NF）

6.2.9 规范化小结

6.2.2 码

- 定义6.4 设K为关系模式R<U,F>中的属性或属性组合。若K $\xrightarrow{F}$ U，则K称为R的一个候选码(Candidate Key)。
  - 如果U部分函数依赖于K，即K $\xrightarrow{P}$ U,则K称为超码（Surpkey）
  - 候选码是最小的超码，即K的任意一个真子集都不是候选码

[例] S(Sno, Sdept, Sage)
Sno→（Sno, Sdept，Sage）， Sno是码
(Sno, Sdept)、 (Sno, Sage)、 (Sno, Sdept, Sage) 是超码
SC(Sno, Cno, Grade)中，(Sno, Cno)是码

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

[例] S(Sno, Sname, Sdept, Sage)， 假设学生无重名
Sno、 Sname是候选码，选择Sno为主码。

6.2.2 码

- 主属性与非主属性
  - 包含在任何一个候选码中的属性，称为**主属性**（Prime attribute）
  - 不包含在任何码中的属性称为**非主属性**（Nonprime attribute）或非码属性（Non-key attribute）

[例] S(Sno, Sdept, Sage)，Sno是码， Sno是主属性， Sdept, Sage是非主属性。
SC(Sno, Cno, Grade)中，(Sno, Cno)是码，
Sno, Cno是主属性， Grade是非主属性

- 全码：整个属性组是码，称为**全码**（All-key）

[例] 关系模式 R（P，W，A） P：演奏者 W：作品 A：听众
语义：一个演奏者可以演奏多个作品，某一作品可被多个演奏者演奏，听众可以欣赏不同演奏者的不同作品
R（P，W，A）码为(P，W，A)，即全码，All-Key。

6.2.2 码：外部码——外码

定义6.5

关系模式 R<U,F>，U中属性或属性组X 并非 R的码，但 X 是另一个关系模式的码，则称 X 是R 的**外部码（Foreign key）也称外码。**

SC（Sno，Cno，Grade）中，Sno不是码，但Sno是关系模式S（Sno，Sdept，Sage）的码，则Sno是关系模式SC的外部码。

- 主码与外部码一起提供了表示关系间联系的手段

思考与讨论

❖ 思考题

已知 关系模式 R<U,F>,
U = {A,B,C,D,E,G}
F = {AC→B，CB→D，A→BE，E→GC}
求关系R的候选码？

❖ 如何根据已知的F, 求一个关系模式R的候选码？

- 简单情况下，可以用观察、验证的方法
- 一般情况下，使用算法

6.2.3 范式

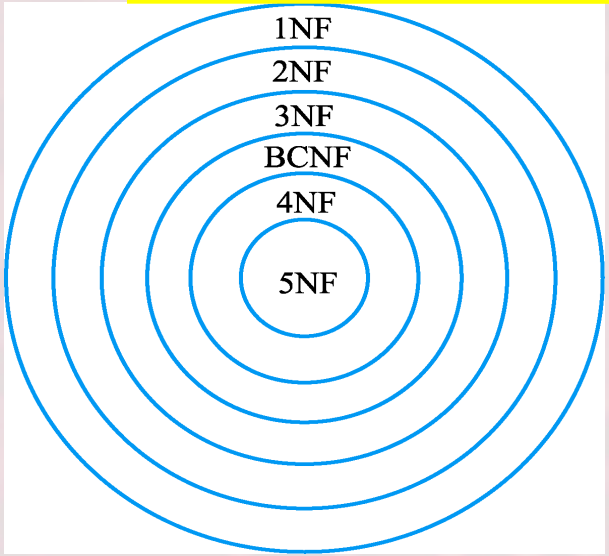
- 范式是符合某一种级别的关系模式的集合。
- 关系数据库中的关系必须满足一定的要求。
  满足不同程度要求的为不同范式。
- 范式的种类：
  - 第一范式(1NF)
  - 第二范式(2NF)
  - 第三范式(3NF)
  - BC范式(BCNF，Boyce和Codd共同提出的范式)
  - 第四范式(4NF)
  - 第五范式(5NF)

范式（续）

❖ 各种范式之间存在联系：

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

■ 某一关系模式R为第n范式，可简记为$R \in nNF$。



一个低一级范式的关系模式，通过模式分解（schema decomposition）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）**。

第一范式

❖ 1NF的定义

如果一个关系模式R的所有属性都是不可分的基本数据项，则R∈1NF。

科目余额表

科目名称	期初余额		本期发生额		期末余额	
	借方	贷方	借方	贷方	借方	贷方
现金	950		4,360	4,350	960	
银行存款	2,690		14,910	7,460	10,140	
应收帐款	16,660		1,740	18,400	0	
原材料	5,000		1,720	2,620	4,100	
预付账款	2,500		5,000	3,500	4,000	
待摊费用	500		200	100	600	
固定资产	5,400		5,000	0	10,4	
短期借款		2,000	2,000			0
应付帐款		3,700	4,400			2,000
应付票据	0	5,000	4,000	2,	0	3,600
预提费用	0	660	0	3	0	1,000
实收资本		20,000			0	20,000
未分配利润		2,340	0			3,600
	33,700	33,700	43,330	0	3	30,200

职工表

编号	职工姓名	性别	出生年月	籍贯	民族	工作时间	技术职务
1	李漱玉	女	09-01-69	北京市	汉族	07-01-90	讲师
2	王清照	女	11-01-51	山东济南	汉族	09-01-99	教授
3	辛如虎	男	08-01-54	河北无极	汉族	12-01-73	副教授
4	柳长亭	男	06-01-60	河南光山	回族	01-01-84	副教授
5	张　煜	男	07-01-72	福建厦门	汉族	07-01-95	教
6	周春花	女	01-01-63	上海市	汉族	10-01-	副教授
7	李　甫	男	02-01-68	山东青岛	汉族	06-0	助理研究员
8	欧阳太白	男	01-01-47	广西桂林	壮族	08-	研究员

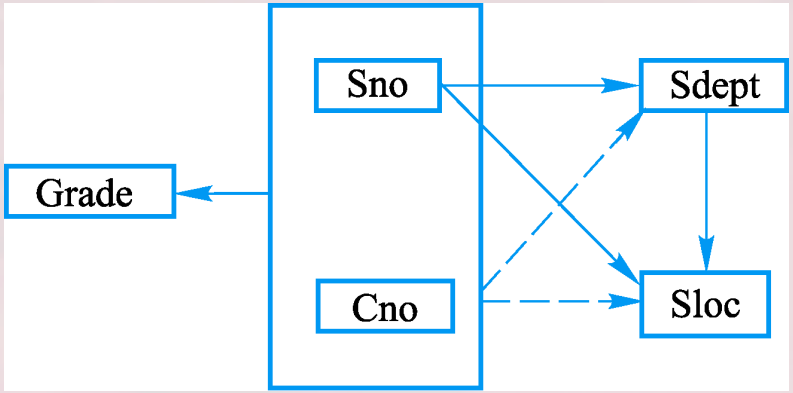
❖ 第一范式是对关系模式的最起码的要求。不满足第一范式的数据库模式不能称为关系数据模式。

6.2.4 第二范式（2NF）

满足第一范式的关系模式并不一定是一个好的关系模式。

[例] 关系模式　S-L-C(Sno, Cno, Sdept, Sloc, Grade)

Sloc为学生住处，假设每个系的学生住在同一个楼。



$(Sno, Cno) \xrightarrow{F} Grade$

$Sno \rightarrow Sdept$,

$(Sno, Cno) \xrightarrow{P} Sdept$

$Sno \rightarrow Sloc$

$(Sno, Cno) \xrightarrow{P} Sloc$

$Sdept \rightarrow Sloc$

1. S-L-C满足第一范式。
2. S-L-C的码为(Sno, Cno)，主属性：Sno, Cno。
   非主属性：Grade ，Sdept和Sloc。
3. 非主属性 Sdept 和 Sloc 部分函数依赖于码(Sno, Cno)。

6.2.4 第二范式（2NF）

❖2NF的定义

定义6.6 若关系模式R∈1NF，并且每一个非主属性都完全函数依赖于R的码，则R∈2NF。

■ 例：

S-L-C(Sno, Cno, Sdept, Sloc, Grade) ∈1NF

S-L-C(Sno, Cno, Sdept, Sloc, Grade) ∉2NF

非主属性 Sdept 和 Sloc 部分函数依赖于码(Sno, Cno)。

6.2.4 2NF（续）

- 一个关系模式R不属于2NF，就会产生问题。
- 例如S-L-C存在的问题：(1) 插入异常

  假设Sno＝2014102，Sdept＝IS，Sloc＝N的学生还未选课，因课程号是主属性，因此该学生的信息无法插入SLC。

Sno	Sdept	Sloc	Cno	Grade
2014101	IS	N	3	89
2014101	IS	N	2	97
2014101	IS	N	5	88
2014103	IS	N	1	86
2014103	IS	N	3	92
2014104	IS	N	3	79
2014102	IS	N	null	null

6.2.4 2NF（续）

(2) 删除异常

假定2014104学生只选修了3号课程这一门课。现在因身体不适，他连3号课程也不选修了。因课程号是主属性，此操作将导致该整个元组的删除。这样，2014104学生信息都被删除了。

Sno	Sdept	Sloc	Cno	Grade
2014101	IS	N	3	89
2014101	IS	N	2	97
2014101	IS	N	5	88
2014103	IS	N	1	86
2014103	IS	N	3	92

6.2.4 2NF（续）

(3) 数据冗余度大

如果一个学生选修了8门课程，那么他的Sdept和Sloc值就要重复存储了8次。

Sno	Sdept	Sloc	Cno	Grade
2014101	IS	N	3	89
2014101	IS	N	2	97
2014101	IS	N	5	88
2014103	IS	N	1	86
2014103	IS	N	3	92
2014104	IS	N	3	79
2014101	IS	N	1	72
2014101	IS	N	4	65
2014101	IS	N	6	99
2014101	IS	N	7	83
2014101	IS	N	8	75

6.2.4 2NF（续）

(4) 修改复杂

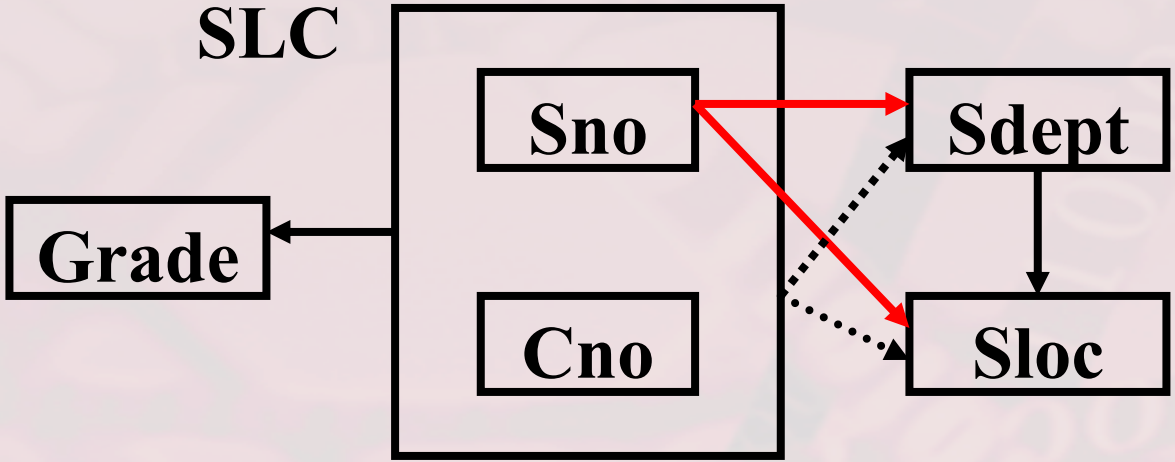
例如学生转系，在修改此学生元组的Sdept值的同时，还可能需要修改住处（Sloc）。如果这个学生选修了K门课，则必须无遗漏地修改K个元组中全部Sdept、Sloc信息。

Sno	Sdept	Sloc	Cno	Grade
2014101	IS	N	3	89
2014101	IS	N	2	97
2014101	IS	N	5	88
2014103	IS	N	1	86
2014103	IS	N	3	92
2014104	IS	N	3	79
2014101	IS	N		
2014101				
2014101				
2014101	IS	N		
2014101	IS	N	8	75

因此，SLC不是一个好的关系模式。

6.2.4 2NF（续）

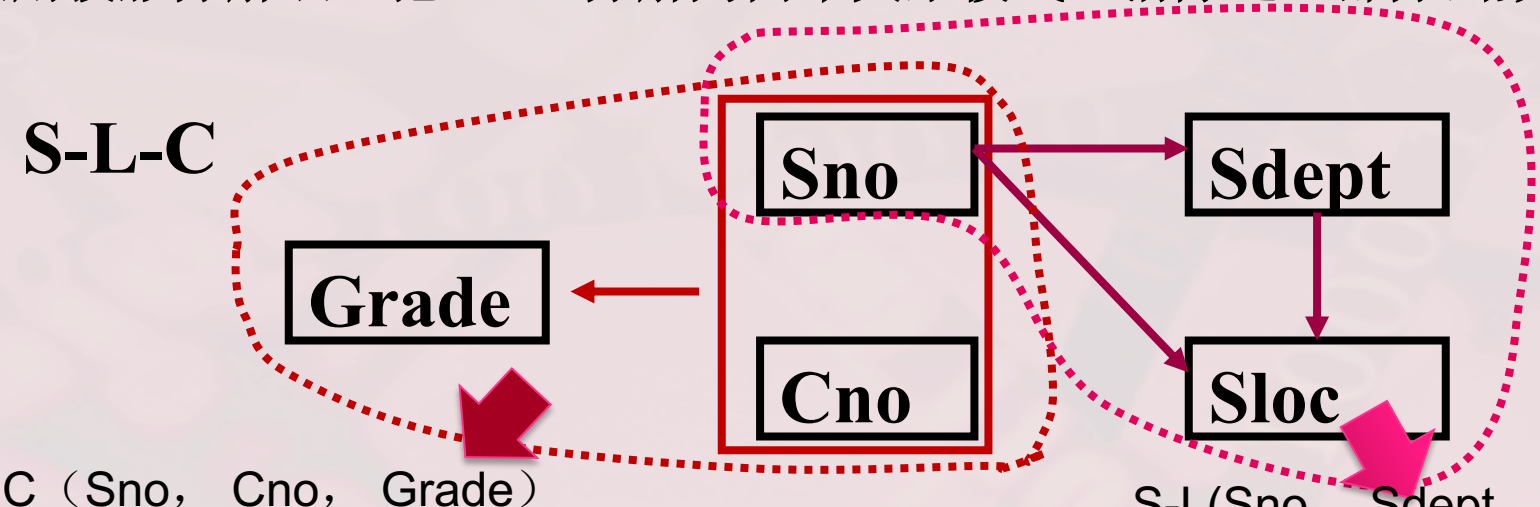
- 原因： SLC(Sno, Sdept, Sloc, Cno, Grade) 中
- Sdept、 Sloc部分函数依赖于码。SLC的码为(Sno, Cno)



6.2.4 2NF（续）

❖ 解决方法

采用投影分解法，把S-L-C分解为两个关系模式，消除这些部分函数依赖



SC（Sno， Cno， Grade）

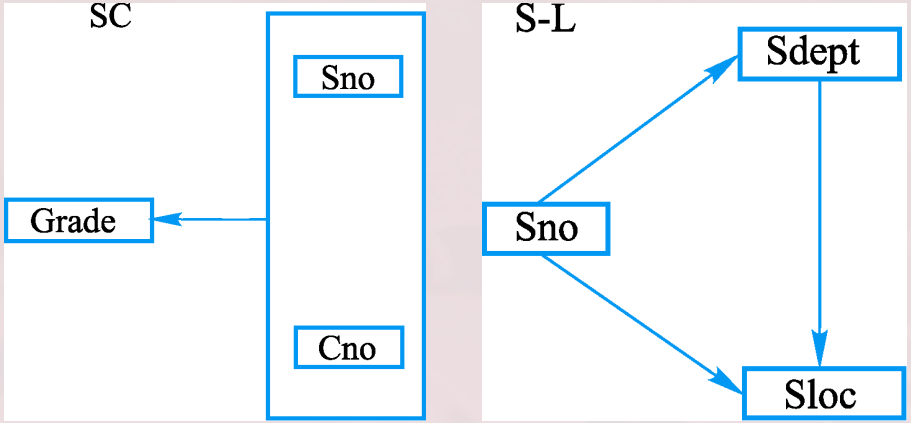
Sno	Cno	Grade

S-L(Sno， Sdept， Sloc)

Sno	Sdept	Sloc

6.2.4 2NF（续）

- 函数依赖图：



- 关系模式SC的码为（Sno, Cno），关系模式S-L的码为Sno
- 非主属性对码都是完全函数依赖了。他们都是2NF。
- 从而使上述四个问题在一定程度上得到了一定的解决。

6.2.4 2NF（续）

SC

Sno	Cno	Grade

S-L

Sno	Sdept	Sloc

(1) 由于学生选修课程的情况与学生的基本情况是分开存储在两个关系中的，在S-L关系中可以插入尚未选课的学生。

(2) 删除一个学生的所有选课记录，只是SC关系中没有关于该学生的记录了，S-L关系中关于该学生的记录不受影响。

(3) 不论一个学生选多少门课程，他的Sdept和Sloc值都只存储1次。这就大大降低了数据冗余。

(4) 学生转系只需修改S-L关系中该学生元组的Sdept值和Sloc值，由于Sdept、 Sloc并未重复存储，因此减化了修改操作。

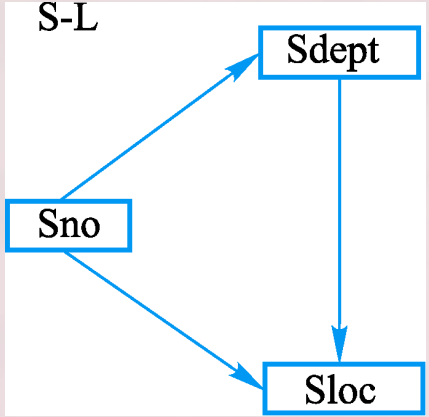
6.2.5 第三范式（3NF）

- **2NF还有什么问题？**
  - 采用投影分解法，把S-L-C分解为两个关系模式：SC和S-L，消除了S-L-C中非主属性对码的部分函数依赖。
  - 一般地，如果把1NF关系模式 通过投影分解方法，消除非主属性对码的部分函数依赖，分解为多个2NF的关系模式。
  - 可以在一定程度上减轻 原1NF关系模式中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。
  - 但是还不能完全消除关系模式中的各种异常情况和数据冗余

6.2.5 第三范式（3NF）

■ 2NF关系模式S-L(Sno, Sdept, Sloc)中

函数依赖：



Sloc传递函数依赖于Sno，即S-L中存在非主属性对码的传递函数依赖Sno $\xrightarrow{传递}$ Sloc 。

6.2.5 第三范式（3NF）

S-L关系存在的问题：

(1) 插入异常

如果某个系因种种原因（例如刚刚成立），目前暂时没有在校学生，我们就无法把这个系的信息，如**MA, S**，存入数据库。

Sno	Sdept	Sloc
2014101	IS	N
2014102	IS	N
2014103	IS	N
2014104	IS	N
null	MA	S

6.2.5 第三范式（3NF）

(2) 删除异常

如果某个系（如IS）的学生全部毕业了，我们在删除该系学生信息的同时，把这个系的信息，如IS, N，也丢掉了。

Sno	Sdept	Sloc
~~2014101~~	~~IS~~	~~N~~
~~2014102~~	~~IS~~	~~N~~
~~2014103~~	~~IS~~	~~N~~
~~2014104~~	~~IS~~	~~N~~
2014105	**PH**	**S**
2014106	**PH**	**S**

6.2.5 第三范式（3NF）

(3) 数据冗余度大

每一个系的学生都住在同一个地方，关于系的住处的信息却重复出现，重复次数与该系学生人数相同。

Sno	Sdept	Sloc
2014101	IS	N
2014102	IS	N
2014103	IS	N
2014104	IS	N
2014105	PH	S
2014106	PH	S
2014107	PH	S
2014108	PH	S
……	……	……

6.2.5 第三范式（3NF）

(4) 修改复杂

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

Sno	Sdept	Sloc
2014101	IS	S
2014102	IS	S
2014103	IS	S
2014104	IS	S
2014105	PH	S
2014106	PH	
2014107		
2014108	PH	S
.....	.....	.....

所以，
S-L仍不是一个好的关系模式。

6.2.5 第三范式（3NF）

❖ 原因:

S-L中Sloc传递函数依赖于Sno，

即：非主属性传递函数依赖码

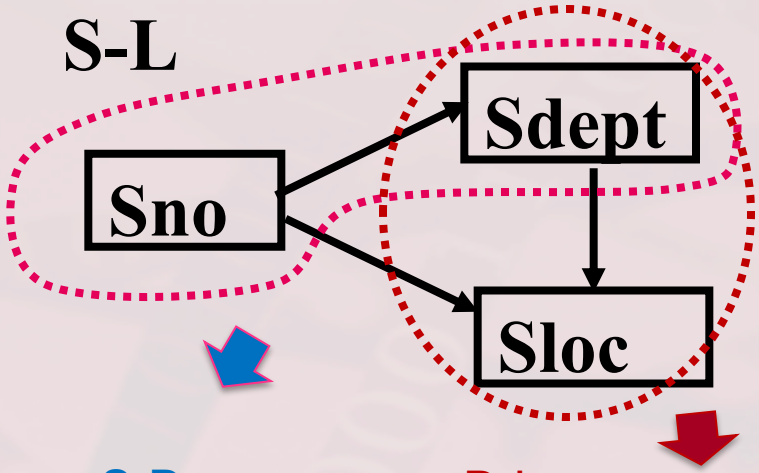
❖ 解决方法

采用投影分解法，把S-L分解为两个关系模式，以消除传递函数依赖：

S-D（Sno， Sdept）

D-L（Sdept， Sloc）

S-D的码为Sno， D-L的码为Sdept



S-D

Sno	Sdept

D-L

Sdept	Sloc

6.2.5 第三范式（3NF）

S-D

Sno	Sdept

D-L

Sdept	Sloc

➢异常的情况得到改善:

(1) D-L关系中可以插入系的信息，即使还没有在校学生。

(2) 某个系的学生全部毕业了，只是删除S-D关系中的相应元组，D-L关系中关于该系的信息仍存在。

(3) 关于系的住处的信息只在D-L关系中存储一次。

(4) 当学校调整某个系的学生住处时，只需修改D-L关系中一个元组的Sloc属性值。

6.2.5 第三范式（3NF）

- S-D的码为Sno， D-L的码为Sdept。

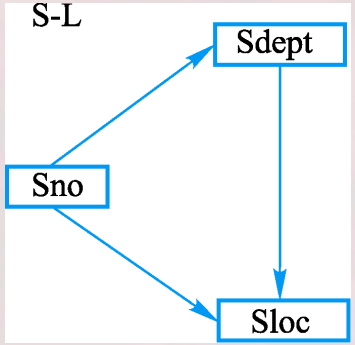
在分解后的关系模式中既没有非主属性对码的部分函数依赖，也没有非主属性对码的传递函数依赖，进一步解决了上述四个问题。

6.2.5 第三范式（3NF）

❖3NF的定义

定义6.7　关系模式R<U，F>∈1NF，若R中不存在这样的码X、属性组Y及非主属性Z（$Y \supsetneq Z$），使得X→Y，Y→Z，$Y \not\to X$，成立，则称R<U，F> ∈ 3NF

例：S-D（Sno， Sdept） ∈ 3NF
　　D-L（Sdept， Sloc）∈ 3NF



S-L 不存在部分函数依赖，但是存在传递函数，所以

S-L(Sno, Sdept, Sloc) ∈ 2NF

S-L(Sno, Sdept, Sloc) ∉ 3NF

6.2.5 第三范式（续）

- 3NF的一些性质：
  - 若R∈3NF，则R的每一个非主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。
  - 如果R∈3NF，则 R∈2NF。
  - 采用投影分解法将一个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。
  - 3NF等价定义：设F是关系模式R的FD集，如果对F中每个非平凡的FD X→Y，都有X是R的超键，或者Y的每个属性都是主属性，则称R是3NF的模式。
  - 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。

一个例子

例：关系模式STJ（S，T，J）中，S表示学生，T表示教师，J表示课程

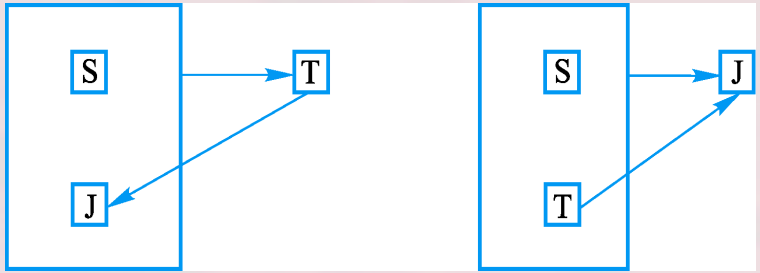
语义：

假设每一教师只教一门课　　T→J

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　　（S，J）→T

某个学生选修某个教师的课就确定了所选课的名称　　(S，T)→J

T→J，（S，J）→T，(S，T)→J



1 候选码：(S，J）和（S，T）

2 S T J 都是主属性

3 不存在非主属性对码的部分函数依赖和传递依赖

3 STJ∈3NF

一个例子

虽然STJ（S,T,J）∈3NF，但它仍存在增删改等异常，还不是一个理想的关系模式。

(1) 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。

(2) 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。

(3) 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。

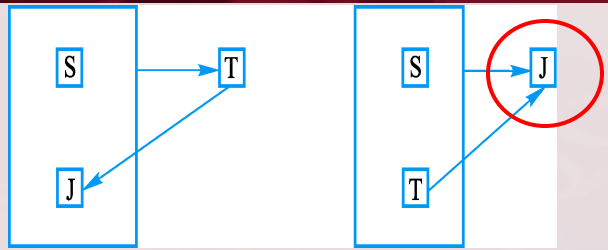
(4) 修改复杂

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。

一个例子

原因：

主属性J部分依赖于码(S, T)。因为T→J



解决方法：

采用投影分解法，将STJ分解为二个关系模式：ST(S，T)；TJ(T，J)

一个例子

- ST的码为（S，T），TJ的码为T。

在分解后的关系模式中没有任何属性对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：

(1) TJ关系中可以存储所开课程尚未有学生选修的教师信息。

(2) 选修某个老师所开设的课程的学生全部毕业了，只是删除ST关系中的相应元组，不会影响TJ关系中相应教师开设该门课程的信息。

(3) 关于每个教师开设课程的信息只在TJ关系中存储一次。

(4) 某个教师开设的某门课程改名后，只需修改TJ关系中的一个相应元组即可。

6.2.6 BC范式（BCNF）

- BCNF（Boyce Codd Normal Form）是由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，有时也称为扩展的第三范式。

- **BCNF的定义**

  定义6.8 设关系模式R<U，F>∈1NF，如果对于R的每个函数依赖X→Y，且$X \supsetneq Y$ 时，X必含有码，那么R∈BCNF。

即，在关系模式**R<U，F>**中，如果每一个决定因素都包含码，则**R∈BCNF**。

例：STJ（S，T，J）∈ 3NF　　**T→J，（S，J）→T，（S，T)→J**

ST（S，T）∈ BCNF　　**ST的码为（S，T），all-key**

TJ（T，J）∈ BCNF　　**TJ的码为T，T→J**

6.2.6 BC范式（BCNF） （续）

- **BCNF的关系模式所具有的性质**

1.所有非主属性对每一个码都是完全函数依赖。

2.所有主属性对每一个不包含它的码也是完全函数依赖。

3.没有任何属性完全函数依赖于非码的任何一组属性。

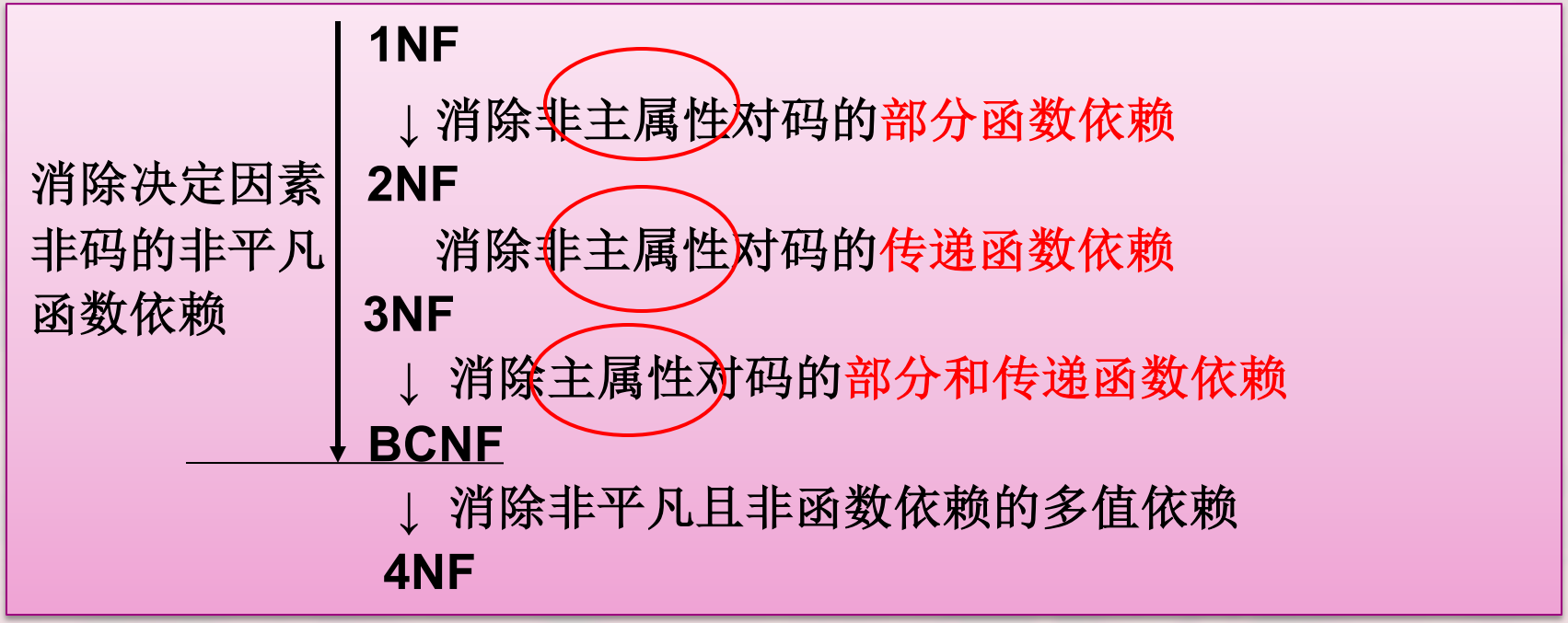
- 如果关系模式R∈BCNF，必定有R∈3NF
- 如果关系模式R∈3NF，不一定有 R∈BCNF
- 如果一个关系数据库中的所有关系模式都属于BCNF，那么在函数依赖范畴内，它已实现了模式的彻底分解，达到了最高的规范化程度，消除了操作异常诸多问题。

6.2.9 规范化小结

- 一个关系模式只要其分量都是不可分的数据项，它就是规范化的关系模式，但这只是最基本的规范化。
- 规范化程度过低的关系模式不一定能够很好地描述现实世界，可能会存在插入异常、删除异常、修改复杂、数据冗余等问题，解决方法就是对其进行规范化，转换成高级范式。
- 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集合，这种过程就叫关系模式的规范化。
- 关系数据库的规范化理论是数据库逻辑设计的工具。

规范化小结（续）

❖ 关系模式规范化的基本步骤



规范化小结（续）

- 规范化的基本思想:
  - 逐步消除数据依赖中不合适的部分，使模式中的各关系模式达到某种程度的“分离”
  - 采用“一事一地”的模式设计原则
    - 让一个关系描述一个概念、一个实体或者实体间的一种联系。
    - 若多于一个概念就把它“分离”出去。
  - 规范化实质上是概念的单一化

举一反三

❖各级范式定义

1）关系模式满足1NF，但不一定满足2NF

2）关系模式满足2NF，则一定满足1NF

3）关系模式不满足2NF，

- 则一定存在“非主属性对候选键的局部依赖”

4）满足3NF，一定满足2NF，也一定满足1NF

5）若关系模式R中没有非主属性，则满足3NF

6）若关系模式R的主码是全码，则满足？

6.3 数据依赖的公理系统

- 数据依赖的公理系统是模式分解算法的理论基础。
- 函数依赖的一个有效而完备的公理系统——Armstrong公理系统，一套推理规则，是模式分解算法的理论基础。

已知：R<U, F>，U={X,C,W,Y, Z, }，F={X→YZ, Z→CW}

问：X→CWYZ是否为F逻辑蕴含

用途

- 从一组函数依赖求得蕴含的函数依赖。例如问$X \rightarrow Y$是否被F所蕴含。
- 求给定关系模式的码。

- 逻辑蕴含

定义6.11 对于满足一组函数依赖F的关系模式$R<U, F>$，其任何一个关系r，若函数依赖$X \rightarrow Y$都成立（即r中任意两元组t，s，若$t[X]=s[X]$，则$t[Y]=s[Y]$），则称F逻辑蕴含$X \rightarrow Y$。

1. Armstrong公理系统

Armstrong公理系统

设U为属性集总体，F是U上的一组函数依赖， 于是有关系模式$R<U, F>$。

对$R<U, F>$来说有以下的推理规则：

- 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。

定理6.1 Armstrong推理规则是正确的

（证明参见《数据库系统概论P.190~P.191》）

2. 导出规则

1.根据A1，A2，A3这三条推理规则可以得到下面三条推理规则：

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

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

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

  （A2， A3） $XW \rightarrow YW$

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

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

2.根据合并规则和分解规则，可得引理6.1

引理6.I $X \rightarrow A_1 A_2 \ldots A_k$成立的充分必要条件是 $X \rightarrow A_i$ 成立（i =I，2，…，k）。

3. 函数依赖闭包

- 闭包 F^+

  定义6.I2 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（closure），记为F^+。

- X关于函数依赖集F的闭包X_F^+

  定义6.13 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。

闭包 F^+

已知关系模式R（ABC），F = { A→B，B→C }，求F^+

求解

– 据已知条件和推理规则，可知F^+有43个FD

A→ Φ	AB→ Φ	AC→ Φ	ABC→ Φ	B→ Φ	C→ Φ
A→A	AB→A	AC→A	ABC→A	B→B	C→C
A→B	AB→B	AC→B	ABC→B	B→C	Φ → Φ
A→C	AB→C	AC→C	ABC→C	B→BC	
A→AB	AB→AB	AC→AB	ABC→AB	BC→ Φ	
A→AC	AB→AC	AC→AC	ABC→AC	BC→B	
A→BC	AB→BC	AC→BC	ABC→BC	BC→C	
A→ABC	AB→ABC	AC→ABC	ABC→ABC	BC→BC	

– 3个属性，F^+ = { 43个FD }；问题规模指数级

3. 函数依赖闭包

❖ 关于闭包的引理

引理6.2 设F为属性集U上的一组函数依赖，X，$Y \subseteq U$，$X \rightarrow Y$能由F根据Armstrong公理导出的充分必要条件是$Y \subseteq X_F^+$。

❖ 引理6.2的用途

1. 将判定$X \rightarrow Y$是否能由F根据Armstrong公理导出的问题，就转化为求出X_F^+，判定Y是否为X_F^+的子集的问题。
2. 如果$X_F^+ = U$，X是R<U，F> 的候选码。

❖ X_F^+可以用算法来求得！

4. 求属性集X 关于F的闭包X_F^+

算法6.I

对于R<U，F>，求属性集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^+

步骤：

迭代

算法6.1

求属性集X（X⊆U）关于U上的函数依赖集F的闭包X_F^+

方法:① 令$X^{(0)}=X$，i=0；

② 对F中的每一个函数依赖$Y\rightarrow Z$，若$Y\subseteq X^{(i)}$，令$X^{(i+1)}=X^{(i)}\cup Z$。

③ 若$X^{(i+1)}\neq X^{(i)}$，则用i+1代替i，转②；

④ 若$X^{(i+1)}=X^{(i)}$，则$X^{(i)}$即为X_F^+

求 X_F^+ 的例子

[例1] 已知关系模式R<U, F>，其中U={A,B,C,D,E}；
F={AB->C,B->D,C->E,EC->B,AC->B}。
求$(AB)_F^+$。

设$X^{(0)}$=AB

计算$X^{(1)}$：　AB->C, B->D

$X^{(1)}$=AB U CD=ABCD

因为$X^{(0)} \neq X^{(1)}$，计算$X^{(2)}$；

求 X_F^+ 的例子

[例1] 已知关系模式R<U, F>，其中U={A,B,C,D,E}；
F={AB->C,B->D,C->E,EC->B,AC->B}。
求$(AB)_F^+$。

设$X^{(0)}$=AB

计算$X^{(1)}$：AB->C, B->D

$X^{(1)}$=AB U CD=ABCD

因为$X^{(0)} \neq X^{(1)}$，计算$X^{(2)}$：C->E, AC->B

$X^{(2)}=X^{(1)}$ U BE=ABCDE

这时$X^{(2)} \neq X^{(1)}$，但$X^{(2)}$已等于全部属性集合

$(AB)_F^+$=ABCDE 。

还有一种情况：如果$X^{(i)}=X^{(i-1)}$，则$X^{(i)}$即为所求的闭包。

问：AB→E能否从F中推出？
YES!

求给定关系模式的码的经验方法

- 若属性A仅出现在所有函数依赖的右部，则它一定不包含在任何候选码中；
- 若属性A仅出现在所有函数依赖的左部，则它一定包含在某个候选码中；
- 若属性A既出现在函数依赖的右部，又出现在左部，则它可能包含在候选码中；
- 在上述基础上求属性闭包。

例：设有关系模式*CTHRSG*（*C*，*T*，*H*，*R*，*S*，*G*），满足下列函数依赖：

C→T　每门课程仅有一位教师讲授

HR→C在任一时间，每个教室只能上一门课程

HT→R在一个时间一位教师只能在一个教室上课

CS→G每个学生的每门课程只有一个成绩

HS→R在一个时间每个学生只能在一个教室听课

求其候选码。

解：求所有候选关键字

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

$(HS)_F^{(0)}=HS$

$(HS)_F^{(1)}=HSR$

$(HS)_F^{(2)}=HSRC$

$(HS)_F^{(3)}=HSRCTG$

$(HS)_F^{(4)}=HSRCTG$

$\therefore$ HS是模式CTHRSG的唯一关键字

5. Armstrong公理系统的有效性与完备性

- 有效性与完备性的含义
  - 有效性：由F出发根据Armstrong公理推导出来的每一个函数依赖一定在F^+中
  - 完备性：F^+中的每一个函数依赖，必定可以由F出发根据Armstrong公理推导出来

- 定理6.2 Armstrong公理系统是有效的、完备的。

（证明参见《数据库系统概论P.192~P.193》）

6. 函数依赖集等价的概念

❖函数依赖集等价定义

定义6.14 如果$G^+=F^+$，就说函数依赖集F覆盖G（F是G的覆盖，或G是F的覆盖），或F与G等价。

两个函数依赖集等价是指它们的闭包等价

7. 最小依赖集

- 也称为极小函数依赖集，最小覆盖

定义6.15　如果函数依赖集F满足下列条件，则称F为一个最小依赖集。

(1) F中任一函数依赖的右部仅含有一个属性。

(2) F中不存在这样的函数依赖X→A，使得F与F-{X→A}等价。

(3) 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。

即F中的函数依赖均不能由F中其他函数依赖导出

F中各函数依赖左部均为最小属性集(不存在冗余属性)

例题

- R<U, F>，U=ABCD,
- 函数依赖集 F = {A→BD，AB→C，C→D}。
- 求：F最小函数依赖集
- 解法步骤：
  - 1.将F中的所有函数依赖的右边化为单一属性
  - 2.去掉F中的所有函数依赖左边的冗余属性
  - 3.去掉F中所有冗余的函数依赖

例题图解（1）

- （1）将F中的所有函数依赖右边化为单一属性

$$F=\{A\rightarrow BD,\ AB\rightarrow C,\ C\rightarrow D\}$$

- $F=\{A\rightarrow B,\ A\rightarrow D,\ AB\rightarrow C,\ C\rightarrow D\}$

例题图解（2）

❖2.去掉F中的所有函数依赖左边的冗余属性

$F=\{A\rightarrow B,\quad A\rightarrow D,\quad AB\rightarrow C,\quad C\rightarrow D\}$

$A^{+}=\{A,B,C,D\}$

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

例题图解（2）

❖2.去掉F中的所有函数依赖左边的冗余属性

$F=\{A\rightarrow B,\quad A\rightarrow D,\quad AB\rightarrow C,\quad C\rightarrow D\}$

$A^+=\{A,B,C,D\}$　　$B^+=\{B\}$

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

例题图解（3）

❖3.去掉F中所有冗余的函数依赖关系

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

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

$A^{+}=\{A,B,C,D\}$

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

6.3 数据依赖的公理系统

1. Armstrong公理系统
2. 导出规则
3. 函数依赖闭包F^+
4. 求属性集X（$X \subseteq U$）关于F的闭包X_F^+
5. Armstrong公理系统的有效性与完备性
6. 函数依赖集等价的概念
7. 最小依赖集或最小覆盖的概念

思考题

1. 已知：R<U, F>，U={X,C,W,Y, Z, }，F={X→YZ, Z→CW}
   试证：X→CWYZ

2. 已知：F={X→YZ, Z→CW, W→V, S→T}
   求：X_F^+

6.4 模式的分解

6.4.1 模式分解的3个定义

6.4.2 分解的无损连接性和保持函数依赖性

6.4.3 模式分解的算法

模式分解的3个定义（续）

- 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

- 什么是模式分解

定义6.16 $R<U, F>$的一个分解是指：$\rho=\{R_1<U_1, F_1>, \ldots, R_n<U_n, F_n>\}$，
其中$U= U_1 \cup U_2 \cup \cdots \cup U_n$，并且没有$U_i \subseteq U_J$，$1 \le i, j \le n$，
F_i是F在U_i上的投影。

定义6.17 函数依赖集合$\{X \rightarrow Y \mid X \rightarrow Y \in F^+ \wedge XY \subseteq U_i\}$ 的一个覆盖 Fi 叫作 F 在属性 Ui 上的投影

相应地将 R 存储在二维表 r 中的数据分散到二维表 r1，r2，…，rn中去，其中 ri是 r在属性集 Ui上的投影。

6.4.1 模式分解的3个定义

- 把低一级的关系模式分解为若干个高一级的关系模式并不是唯一的
- 在这些分解方法中，只有能够保证分解后的关系模式与原关系模式等价的方法才有意义
- 从不同的角度去观察问题，对“等价”的概念形成了3种不同的定义：
  - 分解具有无损连接性（lossless join）
  - 分解要保持函数依赖（preserve functional dependency）
  - 分解既要保持函数依赖，又要具有无损连接性
- 这三个定义是实行模式分解的三条准则。

例子

[例] 对于关系模式S-L（Sno， Sdept， Sloc），S-L中有下列函数依赖：

Sno→Sdept

Sdept→Sloc

传递

Sno→Sloc

- 已知S-L∈2NF，该关系模式存在非主属性对码的传递函数依赖。
- 存在插入异常、删除异常、数据冗余度大和修改复杂的问题。需要分解该关系模式，使成为更高范式的关系模式。
- 下面给出几种不同的分解情况，进行分析，引出模式分解的3个定义。

例子：第一种分解情况

- 第一种分解情况：

将S-L分解为下面三个关系模式：

SN(Sno)

SD(Sdept)

SO(Sloc)

- SN、SD和SO都是规范化程度很高的关系模式。
- 但分解后的关系模式会丢失许多信息

例子：第一种分解情况

假设下面是S-L关系模式的一个关系：

S-L

Sno	Sdept	Sloc
95001	CS	A
95002	IS	B
95003	MA	C
95004	IS	B
95005	PH	B

分解后的关系为：

SN

Sno
95001
95002
95003
95004
95005

SD

Sdept
CS
IS
MA
PH

SO

Sloc
A
B
C

- 但分解后的关系，无法查询95001学生所在的系 或 所住的宿舍了！
- 这种分解是不可取的。丢失了许多信息，丢失了数据之间联系的信息。

例子：第二种分解情况

❖ 第二种分解情况：

将S-L分解为下面二个关系模式：

NL(Sno, Sloc)

DL(Sdept, Sloc)

分解后的关系为：

NL

Sno	Sloc
950001	A
950002	B
950003	C
950004	B
950005	B

DL

Sdept	Sloc
CS	A
IS	B
MA	C
PH	B

S-L

Sno	Sdept	Sloc
95001	CS	A
95002	IS	B
95003	MA	C
95004	IS	B
95005	PH	B

例子：第二种分解情况

对NL和DL关系 进行自然连接的结果为：

NL⋈DL

Sno	Sloc	Sdept
95001	A	CS
95002	B	IS
95002	B	PH
95003	C	MA
95004	B	IS
95004	B	PH
95005	B	IS
95005	B	PH

S-L

Sno	Sdept	Sloc
95001	CS	A
95002	IS	B
95003	MA	C
95004	IS	B
95005	PH	B

NL⋈DL比原来的S-L关系多了三个元组

因此我们也无法知道原来的S-L关系中究竟有哪些元组，

从这个意义上说，此分解仍然丢失了信息。

例子：第三种分解情况

❖ 第三种分解情况：

将S-L分解为下面二个关系模式：　　ND(Sno, Sdept)

NL(Sno, Sloc)

分解后的关系为：

ND

Sno	Sdept
95001	CS
95002	IS
95003	MA
95004	IS
95005	PH

NL

Sno	Sloc
95001	A
95002	B
95003	C
95004	B
95005	B

对ND和NL关系进行自然连接的结果为：

ND ⋈ NL

Sno	Sdept	Sloc
95001	CS	A
95002	IS	B
95003	MA	C
95004	CS	A
95005	PH	B

第三种分解情况没有丢失信息，称为分解“具有无损连接性”。

例子：第三种分解情况

- 但是它存在下面的问题：
  - 例如95001学生由CS系转到IS系，ND关系的(95001, CS)元组和NL关系的(95001, A)元组必须同时进行修改，否则会破坏数据库的一致性。

ND

Sno	Sdept
95001	CS
95002	IS
95003	MA
95004	IS
95005	PH

NL

Sno	Sloc
95001	A
95002	B
95003	C
95004	B
95005	B

IS

B

原因：

- S-L中的函数依赖Sdept→Sloc 既没有投影到关系模式ND上，也没有投影到关系模式NL上。
- 这种分解没有保持原关系模式中的函数依赖。

例子：第四种分解情况

❖ 第四种分解情况：

将S-L分解为下面二个关系模式：

ND(Sno, Sdept)， Sno→Sdept

DL(Sdept, Sloc)， Sdept→Sloc

这种分解保持了函数依赖，称为“具有保持函数依赖性”。

例子：第四种分解情况

❖ 再看第四种分解情况：

将S-L分解为下面二个关系模式：

ND(Sno, Sdept)，Sno→Sdept

DL(Sdept, Sloc)，Sdept→Sloc

Sno	Sdept
950001	CS
950002	IS
950003	MS
950004	IS
950005	PH

Sdept	Sloc
CS	A
IS	B
MA	C
PH	B

ND ⋈ DL

Sno	Sdept	Sloc
95001	CS	A
95002	IS	B
95003	MA	C
95004	IS	B
95005	PH	B

这种分解不仅“具有保持函数依赖性”，还“具有无损连接性”。

例子：分解情况分析

在给出的例子中：

第一种情况：既不具有无损连接性，也未保持函数依赖，

第二种情况：既不具有无损连接性，也未保持函数依赖。

第三种情况：具有无损连接性，但未保持函数依赖。

第四种情况：既具有无损连接性，又保持了函数依赖。

6.4.1 模式分解的3个定义

6.4.2 分解的无损连接性和保持函数依赖性

6.4.3 模式分解的算法

1.具有无损连接性的模式分解

定义6.18

$\rho=\{R_1<U_1, F_1>, \ldots, R_n<U_n, F_n>\}$是 $R<U, F>$的一个分解，若对$R<U, F>$的任何一个关系 r均有 r = r在ρ中各关系模式上投影的自然连接成立，则称分解ρ具有无损连接性。简称ρ为无损分解。

- 只有具有无损连接性的分解才能够保证不丢失信息。
- 无损连接性不一定能解决插入异常、删除异常、修改复杂、数据冗余等问题。

- 算法6.2 判别一个分解的无损连接性。 p.197

  特例：定理6.5

算法6.2：无损连接的测试

输入：一个关系模式R（A_1，A_2，…，A_n），

R上的F以及R的一个分解ρ={R_1, R_2, … R_k}。

输出：确定ρ相对于F是否具有无损分解特性。

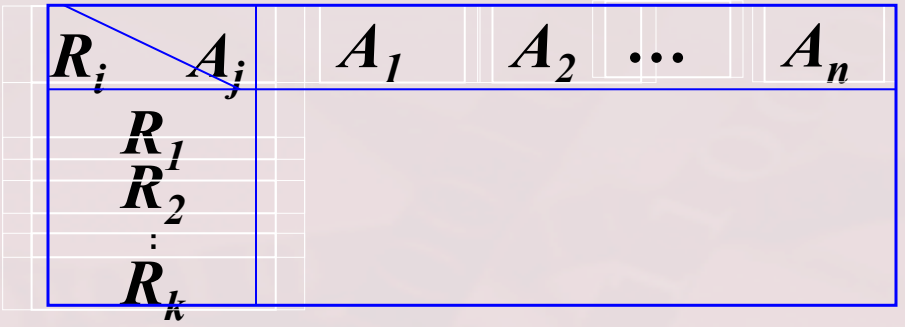
方法：

算法6.2：无损连接的测试

(1) 构造一个k行n列表格

第i行对应于模式R_i，

第j列对应于属性A_j



(2) 填表：若Aj ∈ R_i，

则第i行第j列上填入a_j，否则填入b_{ij}

(3) 修改表：逐一检查F中的每一个函数依赖$X \rightarrow Y$，把X列上符号相同的行对应的Y列的符号改为相同：

Y列有a_j，则将b_{ij}改为a_j；无a_j，则将它们全改为b_{ij}，i一般取其中最小行号

(4) 反复进行(3)直到不能修改

(5) 若某一行变成全a，即$a_1, a_2, \ldots, a_n$，则此分解是无损分解，否则是损失分解。

例： 设有R（U，F），其中，U={A，B，C，D，E}，

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

R 的一个分解为：$\rho=\{R_1(A,D),R_2(A,B),R_3(B,E),R_4(C,D,E),R_5(A,E)\}$

解：构造一个5×5表，并填写该表

A_i / Ri	A	B	C	D	E
R_1：AD	a_1	b_{12}	b_{13} a_3	a_4	b_{15}
R_2：AB	a_1	a_2	b_{23} b_{13} a_3	b_{24} a_4	b_{25}
R_3：BE	b_{31} a_1	a_2	b_{33} b_{13} a_3	b_{34} a_4	a_5
R_4：CDE	b_{41} a_1	b_{42}	a_3	a_4	a_5
R_5：AE	a_1	b_{52}	b_{53} b_{13} a_3	b_{54} a_4	a_5

根据函数依赖集F={ $A\rightarrow C$, $B\rightarrow C$, $C\rightarrow D$, $DE\rightarrow C$, $CE\rightarrow A$ } 修改表

R_3 所在的行为 a_1， a_2， a_3， a_4， a_5，因此，是无损分解。

特殊情况：分解仅由2个模式组成 P197

定理6.5：设R (U,F)，有$\rho=\{R_1, R_2\}$，

则对于F，ρ相对于F是无损分解的

充分必要条件是：

$$U1 \cap U2 \rightarrow U1-U2 \in F+$$

或$$U1 \cap U2 \rightarrow U2-U1 \in F+$$

【例】

设关系模式R（$\{S\#，SN，C\#，G\}$，$\{S\#\rightarrow SN，S\#, C\#\rightarrow G\}$）

的一个分解为：

$\rho=\{R_1（\{S\#, SN\}，\{S\#\rightarrow SN\}）, R_2（\{S\#, C\#, G\}, \{\{S\#, C\#\}\rightarrow G\}）\}$

问：ρ是否无损分解

解：∵ $U_1\cap U_2=S\#$，$U_1-U_2=SN$，

而$S\#\rightarrow SN\in F$

∴ $U_1\cap U_2\rightarrow U_1-U_2$，

分解ρ具有连接不失真性。

2. 保持函数依赖的模式分解

定义6.19

$\rho=\{R_1<U_1, F_1>, \ldots, R_n<U_n, F_n>\}$ 是 $R<U, F>$的一个分解，若F所逻辑蕴含的函数依赖一定也为分解后所有的关系模式中的函数依赖 F_i 所逻辑蕴含，即$F^+ = (F_1 \cup F_2 \cup \ldots \cup F_n)^+$，则称关系模式R的这个分解是保持函数依赖的（Preserve dependency）

$F^+ = (F_1 \cup F_2 \cup \ldots \cup F_n)^+$

引理6.3　给出了判别R的分解是否保持函数依赖的方法。　P.193

保持函数依赖的分解

定义

设F是属性集U上的FD集，Z是U的子集，F在Z上的投影用∏z（F）表示，定义为：

$$\prod_Z(F) = \{ X \rightarrow Y \mid X \rightarrow Y \in F^+ \text{且} XY \subseteq Z \}$$

定义

设$\rho=\{R_1, \ldots, R_k\}$是关系模式$R$的一个分解，F是R上的FD集，

如果有：$\bigcup_{i=1}^{k} \prod_{Ri}(F) \models F$

则称分解ρ保持函数依赖

判断一个分解是否保持依赖

用判断两个函数依赖集是否等价的方法

【例】

设有R（A，B，C，D），$F=\{A\rightarrow B,\ C\rightarrow D\}$，

$\rho=\{R_1(\{A,B\},\{A\rightarrow B\}),R_2(\{C,D\},\{C\rightarrow D\})\}$

问：ρ 是否具有依赖保持性

解：∵ $F=\{A\rightarrow B,\ C\rightarrow D\}$，

$F_1 \cup F_2=\{A\rightarrow B,\ C\rightarrow D\}$

∴ $F^+=(F_1 \cup F_2)^+$

具有依赖保持性

【例】设有R（SNO, BClass, Monitor），其FD集

F = { SNO→BClass，BClass→Monitor }，

分解方案： ρ = { R1, R2 }，其中

R1 = { SNO, Bclass }, F_{R1} = { SNO→Bclass }

R2 = { SNO, Monitor}, F_{R2} = { SNO→Monitor }

问ρ是否具有依赖保持性

解：分解是无损分解，但不保持依赖

∵ $(F_{R1} \cup F_{R2}) \models$ BClass→Monitor ，即$F^+ \not= (F_1 \cup F_2)^+$

依赖保持性的分解 ≠ 连接不失真

一个具有依赖保持性的分解不一定具有连接不失真性。

反之，一个连接不失真分解也不一定具有依赖保持性。

3. 分解的无损连接性和保持函数依赖性

- 如果一个分解具有无损连接性，则它能够保证不丢失信息。
- 如果一个分解保持了函数依赖，则它可以减轻或解决各种异常情况。
- 分解具有无损连接性和分解保持函数依赖是两个互相独立的标准。
  - 具有无损连接性的分解不一定能够保持函数依赖。
  - 保持函数依赖的分解也不一定具有无损连接性。

【例】

设关系模式R（ABC），$\rho=\{AB, AC\}$是R的一个分解。

试分析在$F1=\{A\rightarrow B\}$，$F2=\{A\rightarrow C$，$B\rightarrow C\}$，

$F3=\{B\rightarrow A\}$，$F4=\{C\rightarrow B, B\rightarrow A\}$情况下，

ρ是否具有无损分解和保持依赖的分解特性

相对于$F1=\{A\rightarrow B\}$，　ρ是无损且保持FD的分解

$F2=\{A\rightarrow C$，$B\rightarrow C$，$\}$，　ρ是无损分解，不保持FD

$F3=\{B\rightarrow A\}$，　ρ是损失分解，保持FD

$F4=\{C\rightarrow B, B\rightarrow A\}$，　ρ是损失分解且不保持FD

6.4 模式的分解

6.4.1 模式分解的3个定义

6.4.2 分解的无损连接性和保持函数依赖

6.4.3 模式分解的算法

6.4.3 模式分解的算法

- 算法6.3（合成法）转换为3NF的保持函数依赖的分解
- 算法6.4 转换为3NF既有无损连接性又保持函数依赖的分解
- 算法6.5 （分解法）转换为BCNF的无损连接分解

6.4.3 模式分解的算法

- **按照不同的分解算法，关系模式所能达到的范式是不相同的。**
  - 若要求分解保持函数依赖，那么模式分解一定能够达到3NF，但不一定能够达到BCNF。
  - 若要求分解既具有无损连接性，又保持函数依赖，则模式分解一定能够达到3NF，但不一定能够达到BCNF。
  - 若要求分解具有无损连接性，那么模式分解一定能够达到BCNF。

无损连接	保持依赖	达到的范式
	Y	3NF， 不一定BCNF
Y	Y	3NF， 不一定BCNF
Y		BCNF

模式分解的算法

1.结果为BCNF的无损分解算法　p199算法6.5

输入：R（U，F）

输出：分解ρ=$\{R_1(U_1,F_1),R_2(U_2,F_2),\ldots,R_k(U_k,F_k)\}$

且$F_i=\Pi_{Ri}$（F），R_i满足BCNF。

步骤：

①置初值ρ=$\{R\}$

②如果ρ中所有关系模式都是BCNF，则转④

③如果ρ中有一个关系模式S不是BCNF，

将S中的X不含键且$A\notin X$的$X\rightarrow A$构成S1=XA，

$S2=U_S-A$，用分解{S1，S2}代替S，转②；

④分解结束，输出ρ。

例：将R（city，st，zip）

F={(city，st)→zip, zip→city}

KEY={(city，st),(st,zip)}

决定子不含键

分解为BCNF范式的模式

R（city，st，zip）

分解

ZC(zip,city)

SZ(zip,st)

注：只有分解中每一模式都符合某一范式时，该分解才达到该范式

■ 例：

■ 关系模式R(U, F)，U = { A, B, C }，F = { A→B, BC→A }

- ∵两个候选键：AC、BC
  R满足3NF，∵没有非主属性
- ∴ AC→B
- ∵AC→B，A→B；∴AC→B主属性局部依赖于候选键
- 分解关系模式
  - » R_1(A, C)
  - » R_2(A, B)
- 解决了问题，但丢失FD BC→A，可能会有新问题

例：设有关系模式*CTHRSG*（*C*，*T*，*H*，*R*，*S*，*G*），满足下列

函数依赖：

C→T 每门课程仅有一位教师讲授

HR→C 在任一时间，每个教室只能上一门课程

HT→R 在一个时间，一位教师只能在一个教室上课

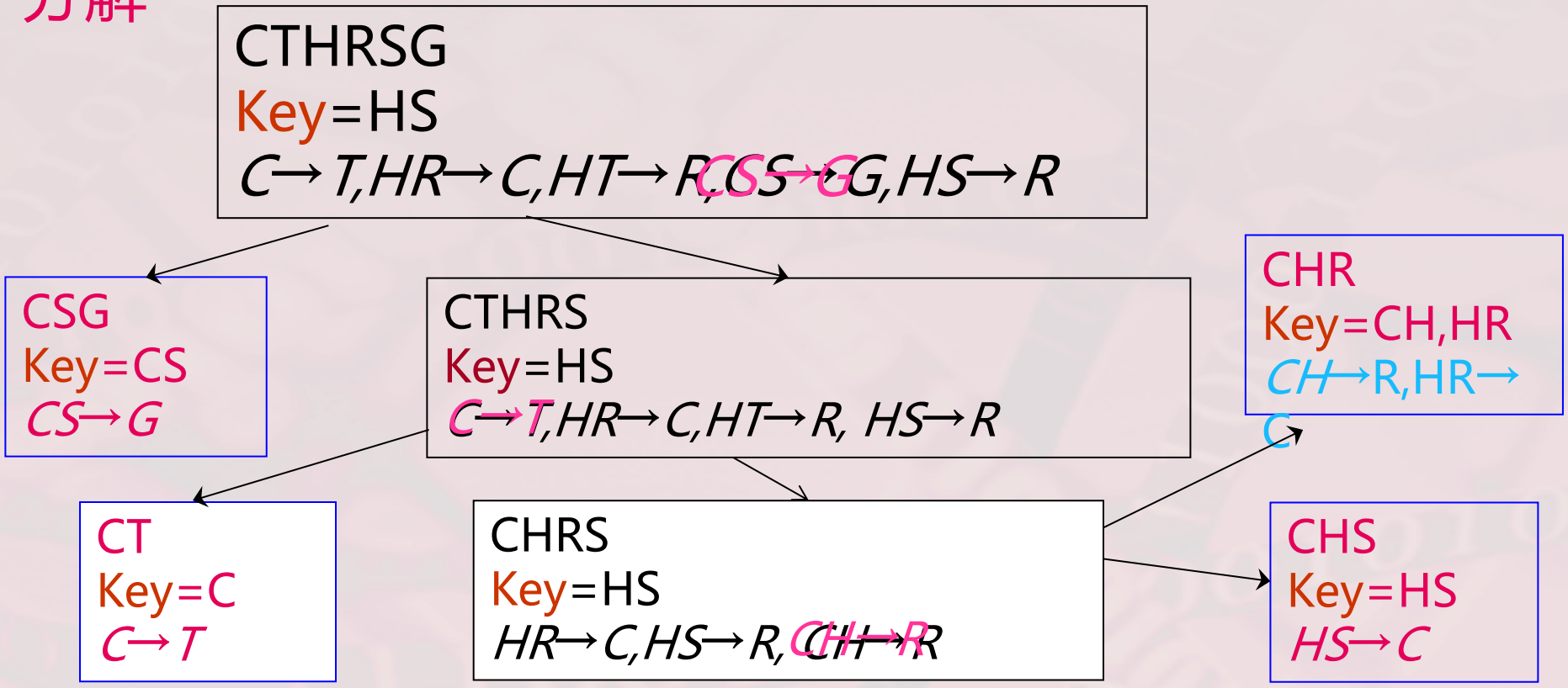
CS→G 每个学生的每门课程只有一个成绩

HS→R 在一个时间，每个学生只能在一个教室内听课

F={*C→T*，*HR→C*，*HT→R*，*CS→G*，*HS→R*}

解：候选关键字*HS*

分解


CTHRSG
Key=HS
C→T,HR→C,HT→R,CS→G,HS→R
CSG
Key=CS
CS→G
CTHRS
Key=HS
C→T,HR→C,HT→R, HS→R
CHR
Key=CH,HR
CH→R,HR→C
CT
Key=C
C→T
CHRS
Key=HS
HR→C,HS→R,CH→R
CHS
Key=HS
HS→C

∴ CTHRSG $\xrightarrow{\text{分解}}$ CSG，CT，CHR，CHS

CSG：学生的各门课程成绩；

CT： 各门课的任课教师；

CHR：每门课程的上课时间和每个时间的上课教室；

CHS：每个学生的上课时间表

？ 函数依赖HT→R

P198 算法6.3

2. 结果为3NF保持依赖的分解算法： R(U,F)

- 不在F中的属性构成一个关系模式
- 将F中每一个X→A构成一个关系模式
- 若F中有X→A1，X→A2， X→An

  用XA1A2…An代替n个XA1，XA2，… ，XAn

例1：CTHRSG(C,T,H,R,S,G)

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

$$\rho=\{CT，CHR，HRT，CSG，HSR\}$$

P199 算法6.4

3. 结果为3NF，且具有保持依赖和无损连接的分解算法：

设ρ={R_1,R_2,…,R_k}是由结果为3*NF*的保持依赖分解算法得到的3*NF*分解，*X*为*R*的一个候选键， 则τ=ρ∪{X}是*R*的一个分解，且τ中的所有关系模式均满足3*NF*，同时，既具有连接不失真性，又具有依赖保持性。

例：CTHRSG(C,T,H,R,S,G)

ρ={*CT*，*CHR*，*HRT*，*CSG*，*HSR*} ∪ {*HS*}

ρ={*CT*，*CHR*，*HRT*，*CSG*，*HSR*}

第六章 关系数据理论

6.1 问题的提出—为什么要学习关系数据理论
6.2 规范化
6.3 数据依赖的公理系统（简单介绍）
6.4 模式的分解（简单介绍）
6.5 小结

小结

- 函数依赖
  - 平凡函数依赖与非平凡函数依赖
  - 完全函数依赖与部分函数依赖
  - 传递函数依赖
  - 码
- 范式的概念
- 关系模式规范化的基本步骤
- **Armstrong公理系统**
- 模式的分解

小结

❖ 关系模式规范化及基本步骤

消除决定属性集非码的非平凡函数依赖

1NF

↓ 消除非主属性对码的部分函数依赖

2NF

↓ 消除非主属性对码的传递函数依赖

3NF

↓ 消除主属性对码的部分和传递函数依赖

BCNF

↓ 消除非平凡且非函数依赖的多值依赖

4NF

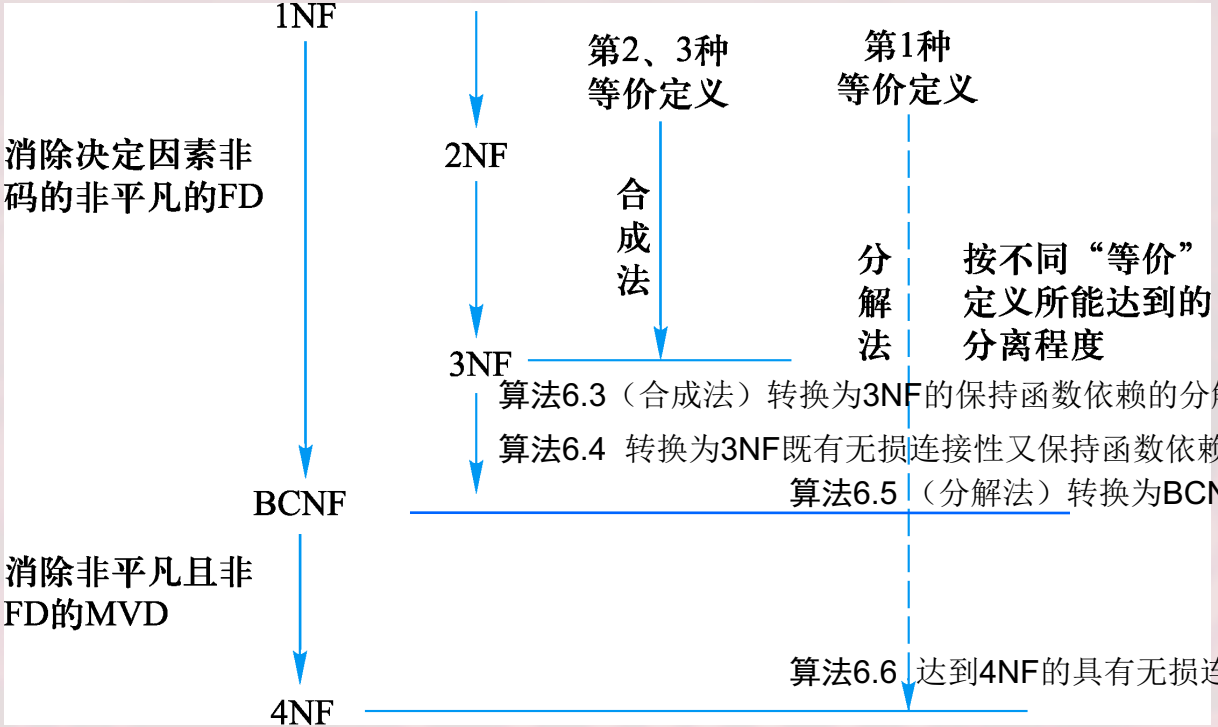
小结

- Armstrong公理系统
  - 自反律；增广律；传递律
  - 合并规则；伪传递规则；分解规则
  - 函数依赖闭包F^+ 求属性集X关于F的闭包X_F^+
  - 最小依赖集

假设。

小结

关系模式分解：模式分解等价的3个准则、模式分解的算法



假设。

小结

关系模式的规范化基本思想

1．逐步消除不合适的数据依赖中使模式中的各关系模式达到某种程度的“分离”——模式分解

2. 一事一地的模式设计原则

3．规范化理论为数据库设计提供了理论指南和算法工具。

4．结合应用环境的具体情况，合理地设计数据库模式。

学习关系规范化理论的重要性：

- 是关系数据库的重要理论基础。
- 为数据库设计提供理论指南和算法工具

—规范化理论研究的实际应用价值。