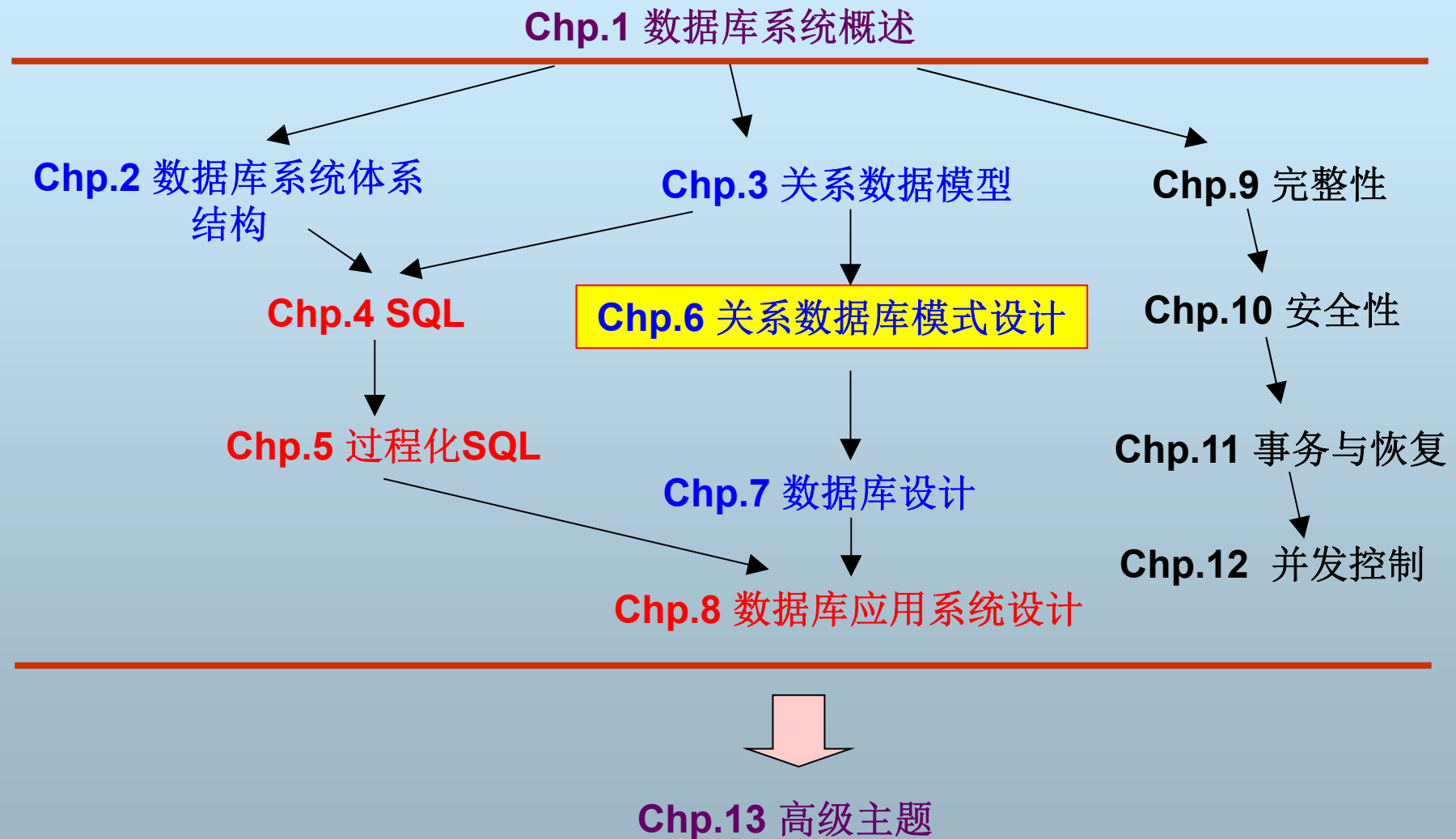


第6章 关系数据库模式设计

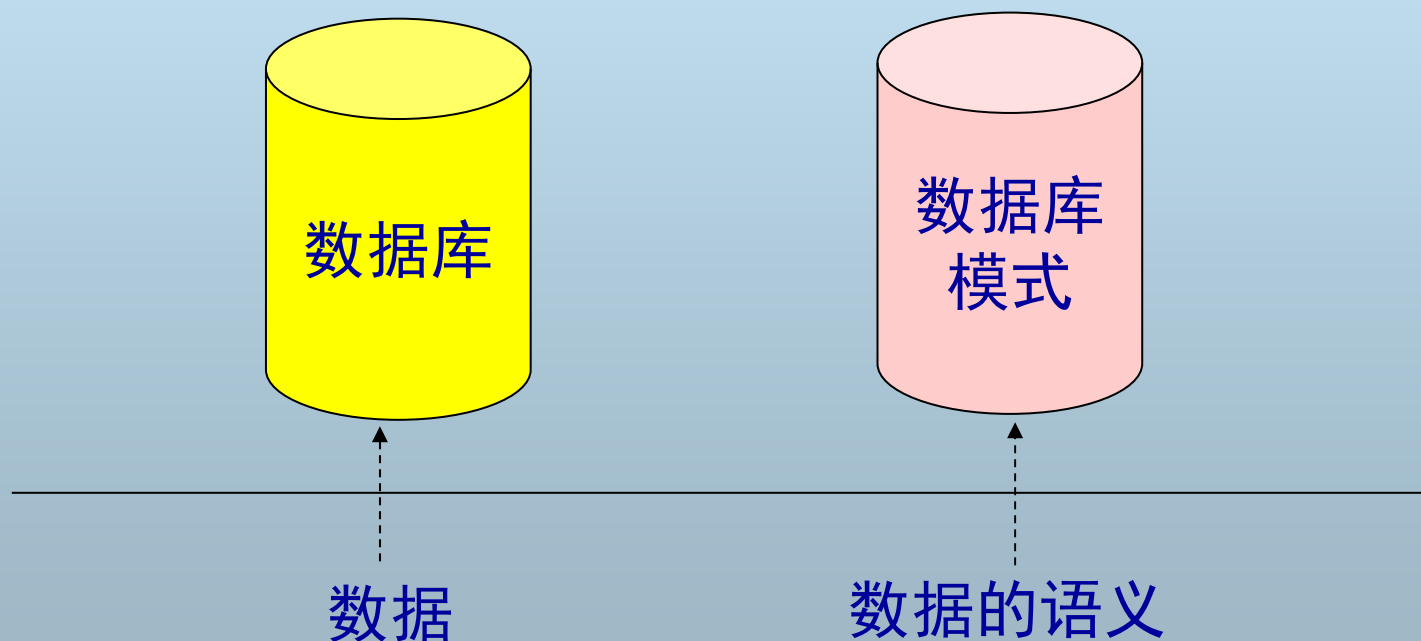


课程知识结构



数据库模式

- 数据库模式是数据库中全体数据的逻辑结构和特征的描述



举例

学号	姓名	年龄
001	张三	20
002	李四	21
003	王五	22



学生(学号:char, 姓名:char, 年龄:int)

模式

数据库

问题的提出

- 如何把现实世界表达成数据库模式？
- 针对一个具体应用，应该如何构造一个**适合**于它的数据库模式？
- 这是数据库的逻辑设计问题——关系数据库的模式设计理论是数据库逻辑设计的理论基础

本章主要内容

- 关系模式的设计问题
- 函数依赖
- 关系模式的分解
- 关系模式的范式

一、关系模式的设计问题

■ 关系模式设计不规范会带来一系列的问题

- 数据冗余
- 更新异常
- 插入异常
- 删除异常

示例关系模式R

示例关系模式 **R(Tname, Addr, C#, Cname)**

一个教师只有一个地址和一个联系电话

一个教师可教多门课程

一门课程只有一个任课教师

因此**R**的主码是 (**C#**)

Tname	Addr	Tel	<u>C#</u>	Cname
T1	A1	6360111	C1	N1
T1	A1	6360111	C2	N2
T1	A1	6360111	C3	N3
T2	A2	6360222	C4	N4
T2	A2	6360222	C5	N5
T3	A3	6360333	C6	N6

1、问题（1）：数据冗余

- 教师T1教了三门课程，他的地址和电话被重复存储了2次

Tname	Addr	Tel	<u>C#</u>	Cname
T1	A1	6360111	C1	N1
T1	A1	6360111	C2	N2
T1	A1	6360111	C3	N3
T2	A2	6360222	C4	N4
T2	A2	6360222	C5	N5
T3	A3	6360333	C6	N6

2、问题（2）：更新异常

- 如果**T1**的地址变了，则需要改变**3**个元组的地址；若有一个未更改，就会出现数据不一致。但**DBMS**无法获知这种不一致

Tname	Addr	Tel	<u>C#</u>	Cname
T1	A1	6360111	C1	N1
T1	A1	6360111	C2	N2
T1	A1	6360111	C3	N3
T2	A2	6360222	C4	N4
T2	A2	6360222	C5	N5
T3	A3	6360333	C6	N6

3、问题（3）：插入异常

- 如果要增加一名教师，但他还未带课，则C#和Cname为空，但由于C#是主码，为空违反了实体完整性，所以这名教师将无法插入到数据库中

Tname	Addr	Tel	<u>C#</u>	Cname
T1	A1	6360111	C1	N1
T1	A1	6360111	C2	N2
T1	A1	6360111	C3	N3
T2	A2	6360222	C4	N4
T2	A2	6360222	C5	N5
T3	A3	6360333	C6	N6

4、问题（4）：删除异常

- 如果教师**T3**现在不带课了，则需将**T3**的元组删去，但同时也把他的姓名和地址电话信息删掉了

Tname	Addr	Tel	<u>C#</u>	Cname
T1	A1	6360111	C1	N1
T1	A1	6360111	C2	N2
T1	A1	6360111	C3	N3
T2	A2	6360222	C4	N4
T2	A2	6360222	C5	N5
T3	A3	6360333	C6	N6

5、如何解决？

■ 方法：模式分解

● 方法1：R分解为

◆ R1(Tname, Addr, Tel)

◆ R2(C#, Cname)

授课信息丢失了

● 方法2

◆ R1(Tname, Addr, Tel, C#)

◆ R2(C#, Cname)

R1中问题依然存在

● 方法3

◆ R1(Tname, Addr, Tel)

◆ R2(Tname ,C#, Cname)

基本解决问题，但又带来联接查询代价

5、如何解决？

- 到底什么样的模式才最佳？怎么分解才能达到要求？标准是什么？如何实现？



关系数据库的模式
式设计理论

二、函数依赖

- 什么是函数依赖
- 码的定义
- 最小函数依赖集

回顾：关系模式的形式化定义：

R (U, D, dom, F)

R为关系模式名，**U**是一个属性集，**D**是**U**中属性的值所来自的域，**Dom**是属性向域的映射集合，**F**是属性间的依赖关系

1、什么是函数依赖？

- 函数依赖是指一个关系模式中一个属性集和另一个属性集间的多对一关系
 - 例如选课关系**SC(S#, C#, Score)**
 - 存在由属性集{**S#, C#**}到属性集{**Score**}的函数依赖
 - ◆ 对于任意给定的**S#**值和**C#**值，只有一个**Score**值与其对应
 - ◆ 反过来，可以存在多个**S#**值和**C#**值，它们对应的**Score**值相等

2、基本概念

- 函数依赖（**FD, Functional Dependency**）的形式化定义
 - 设关系模式 **$R(A_1, A_2, \dots, A_n)$** 或简记为 **$R(U)$** ， **X** 和 **Y** 是 **U** 的子集。 **r** 是 **R** 的任意一个实例（关系），若 **r** 的任意两个元组 **t_1** 、 **t_2** ，由 **$t_1[X] = t_2[X]$** 可导致 **$t_1[Y] = t_2[Y]$** ，即如果 **X** 相等则 **Y** 也相等，则称 **Y** 函数依赖于 **X** 或称为 **X** 函数决定 **Y** ，记作 **$X \rightarrow Y$**
 - ◆ 即 **R** 的 **X** 属性集上的值可唯一决定 **R** 的 **Y** 属性集上的值
 - ◆ 也即对于 **R** 的任意两个元组， **X** 上的值相等，则 **Y** 上的值也必相等
 - **FD**是相对于关系模式而言的，因此关系模式 **R** 的所有实例都要满足**FD**

2、基本概念

■ 例如

- **Student**关系模式中， $\{S\# \} \rightarrow \{Sname\}$ （单个属性可去掉括号，简写成 $S\# \rightarrow Sname$ ）

- **SC**关系模式中， $\{S\#,C\# \} \rightarrow \{Score\}$

■ **FD**是否成立，唯一办法是仔细考察应用中属性的含义。**FD**实际上是对现实世界的断言。数据库设计者在设计时把应遵守的函数依赖通知**DBMS**，则**DBMS**会自动检查关系的合法性

- 对于关系模式 **R(Tname, Addr, C#, Cname)**

- ◆ 若一门课只能有一个教师，则有 $\{C\# \} \rightarrow \{Tname\}$
- ◆ 若一门课可有多个教师任教，则 $\{C\# \} \rightarrow \{Tname\}$ 不成立
- ◆ 因此**FD**是与具体应用相关的

2、基本概念

■ 关系模式的形式化定义：

● **$R(U, D, Dom, F)$**

◆ **R** 为关系模式名， **U** 是一个属性集， **D** 是 **U** 中属性的值所来自的域， **Dom** 是属性向域的映射集合， **F** 是属性间的依赖关系

● **FD** 是关系模式的一部分

3、平凡FD和不平凡FD

- 模式设计的首要问题是确定关系模式的最小函数依赖集
 - 给定一个函数依赖集S，若能找到一个远小于S的函数依赖集T，则DBMS只要实现T就可实现S中的所有函数依赖
- 平凡FD和不平凡FD
 - $X \rightarrow Y$ ，且 $Y \subseteq X$ ，则 $X \rightarrow Y$ 是平凡FD，否则是不平凡FD
- 平凡FD没有什么实际意义，消除平凡FD是缩小函数依赖集大小的一个简单方法

4、函数依赖集的闭包

■ 函数依赖的逻辑蕴含

- 设 F 是关系模式 R 的一个函数依赖集， X 和 Y 是 R 的属性子集，若从 F 的函数依赖中能推出 $X \rightarrow Y$ ，则称 F 逻辑蕴含 $X \rightarrow Y$ ，记作 $F \models X \rightarrow Y$

■ 函数依赖集的闭包

- 被函数依赖集 F 逻辑蕴含的函数依赖的全体构成的集合称为 F 的闭包，记做 F^+

(1) 函数依赖的推理规则

- **Armstrong公理**，可以从给定的函数依赖中推出新的函数依赖
 - 自反律 (**Reflexivity**) : 若 $B \subseteq A$, 则 $A \rightarrow B$ 成立
 - 增广律 (**Augmentation**) : 若 $A \rightarrow B$, 则 $AC \rightarrow BC$ (AC 表示 $A \cup C$)
 - 传递律 (**Transitivity**) : 若 $A \rightarrow B$, $B \rightarrow C$, 则 $A \rightarrow C$
 - 自含律 (**Self_Determination**) : $A \rightarrow A$
 - 分解律 (**Decomposition**) : 若 $A \rightarrow BC$, 则 $A \rightarrow B$, 且 $A \rightarrow C$
 - 合并律 (**Union**) : 若 $A \rightarrow B$, $A \rightarrow C$, 则 $A \rightarrow BC$
 - 复合律 (**Composition**) : 若 $A \rightarrow B$, $C \rightarrow D$, 则 $AC \rightarrow BD$

(1) 函数依赖的推理规则

- $R(A, B, C, D, E, F)$
- $F = \{A \rightarrow BC, B \rightarrow E, CD \rightarrow EF\}$
- $AD \rightarrow F$ 对于函数依赖集 F 是否成立?
 - $A \rightarrow BC$ (已知)
 - $A \rightarrow C$ (分解律)
 - $AD \rightarrow CD$ (增广律)
 - $CD \rightarrow EF$ (已知)
 - $AD \rightarrow EF$ (传递律)
 - $AD \rightarrow F$ (分解律)

(2) 码的形式化定义

- 设关系模式 $R(U)$, F 是 R 的一个FD集, X 是 U 的一个子集, 若
 - $X \rightarrow U \in F^+$, 则 X 是 R 的一个超码, 如果同时
 - 不存在 X 的真子集 Y , 使得 $Y \rightarrow U$ 成立, 则 X 是 R 的一个候选码
- $R(Tname, Addr, C\#, Cname)$
 - $F = \{Tname \rightarrow Addr, C\# \rightarrow Cname, C\# \rightarrow Tname\}$
 - $C\# \rightarrow \{Tname, Addr, C\#, Cname\}$
 - 所以 $C\#$ 是候选码, 若 $C\# \rightarrow Tname$ 不成立, 则候选码为 $\{Tname, C\#\}$

5、属性集的闭包

- 函数依赖集的闭包计算很麻烦
- 给定一个函数依赖集F，如何判断函数依赖 $X \rightarrow Y$ 是否可以从F中推出？
- 属性集的闭包
 - 设F是属性集U上的一个FD集，X是U的子集，则称所有用Armstrong推理规则推出的函数依赖 $X \rightarrow A$ 中所有A的集合，称为属性集X关于F的闭包，记做 X^+
- $X \rightarrow Y$ 能由Armstrong推理规则推出的充要条件是 $Y \subseteq X^+$

(1) 例子

- 关系模式 $R(A, B, C, D)$
- $F = \{A \rightarrow B, B \rightarrow C, B \rightarrow D, A \rightarrow D\}$
 - $A^+ = ABCD$
 - $B^+ = BCD$
 - $C^+ = C$
 - $D^+ = D$
- 不用计算 F^+ , 就可知 $A \rightarrow CD \in F^+$

6、最小函数依赖集

■ 函数依赖集的等价和覆盖

- 设**S1**和**S2**是两个函数依赖集，若 **$S1^+ \subseteq S2^+$** ,则称**S2是S1的覆盖（或S2覆盖S1）**
 - ◆ **DBMS**只要实现**S2**中的函数依赖，就自动实现了**S1**中的函数依赖
- 若**S2是S1的覆盖，且S1是S2的覆盖，则称S1与S2等价**
 - ◆ **DBMS**只要实现任意一个**FD**集，就可自动实现另一个**FD**集

(1) 定义

- 当且仅当函数依赖集F满足下面条件时，F是最小函数依赖集：
 - F的每个FD的右边只有一个属性
 - **F不可约**：F中的每个 $X \rightarrow Y$ ， $F - \{X \rightarrow Y\}$ 与F不等价
 - **F的每个FD的左部不可约**：删除左边的任何一个属性都会使F转变为一个不等价于原来的F的集合

(2) 举例

■ Student(S#,Sname,Age, Sex)

- $F1 = \{S\# \rightarrow Sname, S\# \rightarrow age, S\# \rightarrow sex\}$ 是最小函数依赖集
- $F2 = \{S\# \rightarrow \{S\#, Sname\}, S\# \rightarrow age, S\# \rightarrow sex\}$ 不是最小函数依赖集 【右边不是单属性】
- $F3 = \{S\# \rightarrow Sname, \{S\#, Sname\} \rightarrow age, S\# \rightarrow sex\}$ 不是最小函数依赖集 【左部可约】
- $F4 = \{S\# \rightarrow S\#, S\# \rightarrow Sname, S\# \rightarrow age, S\# \rightarrow sex\}$ 不是最小函数依赖集 【FD可约】

(3) 求最小函数依赖集

- $R(A,B,C,D)$, $F=\{A\rightarrow BC, B\rightarrow C, A\rightarrow B, AB\rightarrow C, AC\rightarrow D\}$
 - 将右边写出单属性并去除重复FD（分解律）
 - ◆ $F=\{A\rightarrow B, A\rightarrow C, B\rightarrow C, A\rightarrow B, AB\rightarrow C, AC\rightarrow D\}$
 - ◆ $F=\{A\rightarrow B, A\rightarrow C, B\rightarrow C, AB\rightarrow C, AC\rightarrow D\}$
 - 消去左部冗余属性
 - ◆ $A\rightarrow C$, $AC\rightarrow D$ 可推出 $A\rightarrow AC$, $A\rightarrow D$, 因此可去除 $AC\rightarrow D$ 中的C
 - ◆ $A\rightarrow C$, 可推出 $AB\rightarrow BC$ 可得 $AB\rightarrow C$, 所以 $AB\rightarrow C$ 中的B是冗余属性
 - ◆ $F=\{A\rightarrow B, A\rightarrow C, B\rightarrow C, A\rightarrow C, A\rightarrow D\}$
 $=\{A\rightarrow B, A\rightarrow C, B\rightarrow C, A\rightarrow D\}$
 - 消去冗余函数依赖
 - ◆ $A\rightarrow C$ 冗余, 因为可由 $A\rightarrow B$, $B\rightarrow C$ 推出
 - ◆ $F=\{A\rightarrow B, B\rightarrow C, A\rightarrow D\}$

三、模式分解

- 概念
- 无损连接(Lossless Join)
- 保持函数依赖(Preserve Dependency)

1、模式分解的概念

- 设有关系模式 $R(U)$ 和 $R_1(U_1)$, $R_2(U_2)$, ..., $R_k(U_k)$, 其中 $U = U_1 \cup U_2 \dots \cup U_k$, 设 $\rho = \{R_1, R_2, \dots, R_k\}$, 则称 ρ 为 R 的一个分解
- 模式分解的含义
 - 属性集的分解
 - 函数依赖集的分解
 - ◆ $R(A, B, C)$, $F = \{A \rightarrow B, C \rightarrow B\}$, 则分解为 $R_1(A, B)$, $R_2(A, C)$ 丢失了 $C \rightarrow B$

2、模式分解的标准

- 具有无损连接
- 要保持函数依赖
- 既具有无损连接，又要保持函数依赖

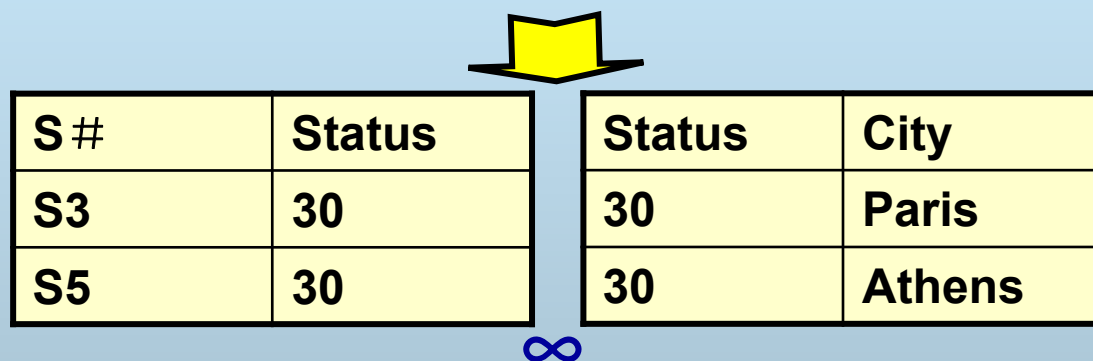
3、无损连接

- 动机
- 概念
- 无损连接的测试

(1) 动机

S #	Status	City
S3	30	Paris
S5	30	Athens

- 模式分解的过程应是可逆的，**R**的所有数据在分解后应没有丢失



S #	Status	City
S3	30	Paris
S3	30	Athens
S5	30	Paris
S5	30	Athens

信息丢失
分解后要不能得到**S3**的
City是**Paris**

(2) 概念

- 设 R 是关系模式，分解成关系模式 $\rho = \{R_1, R_2, \dots, R_k\}$ ， F 是 R 上的一个FD集，若对 R 中满足 F 的每个关系 r ，都有：
$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \dots \bowtie \pi_{R_k}(r)$$
，则称这个分解 ρ 相对于 F 是“无损连接分解”
 - R 的每个关系 r 是它在 R_i 上的投影的自然连接
 - 无损连接保证 R 分解后还可以通过 R_i 恢复

(2) 概念

- 我们记 $m_{\rho}(r) = \bigcap_{i=1}^k \pi_{R_i}(r)$
- 则对于关系模式 **R** 关于 **F** 的无损连接条件是 $r = m_{\rho}(r)$

(2) 概念

R

S #	Status	City
S3	30	Paris
S5	30	Athens

R1

S #	Status
S3	30
S5	30

R2

Status	City
30	Paris
30	Athens



S #	Status	City
S3	30	Paris
S3	30	Athens
S5	30	Paris
S5	30	Athens

$$r \neq m_p(r)$$

所以不是无损连接

(1) Select * From R

(2) Select * From R1,R2

where

R1.Status=R2.Status

返回结果不一致

$$m_{\rho}(r) = \pi_{R1}(r) \bowtie \pi_{R2}(r)$$

(3) 无损连接的测试

■ 方法1: Chase

- 输入：关系模式 $R(A_1, A_2, \dots, A_n)$ ， R 上的函数依赖集 F ， R 的一个分解 $p = \{R_1, \dots, R_k\}$
- 输出：判断 p 相对于 F 是否具有无损连接性
- 算法：Chase

Jeffrey. D. Ullman, Jennifer Widom. A First Course in Database Systems
(岳丽华 金培权 等译. 数据库系统基础教程, 机械工业出版社, 2003)

1) Chase过程

- 构造一个 k 行 n 列的表格，每行对应一个模式 R_i ($1 \leq i \leq k$)，每列对应一个属性 A_j ($1 \leq j \leq n$)，若 A_j 在 R_i 中，则在表格的第 i 行第 j 列处填上 a_j ，否则填上符号 b_{ij}
- 检查 F 的每个FD，并修改表格中的元素，方法如下：
 - 对于 F 中的函数依赖 $X \rightarrow Y$ ，若表格中有两行在 X 分量上相等，在 Y 分量上不相等，则修改 Y ：
 - ◆ 若 Y 的分量中有一个 a_j ，则另一个也修改为 a_j ；
 - ◆ 如果没有 a_j ，则用其中一个 b_{ij} 替换另一个符号（ i 是所有 b 中最小的行数），一直到表格不能修改为止
- 若修改后，表格中有一行是全 a ，即 $a_1a_2 \dots a_n$ ，则 p 相对于 F 是无损连接的分解，否则不是

1) Chase过程

- 扫描一次F后，若表格中未出现全a的行，则进行下一次扫描
 - 由于每次扫描F至少能减少一个符号，而符号有限，因此算法最后必然终止
 - 终止条件
 - ◆ 全a行
 - ◆ 表格扫描后不再发生任何修改

2) Chase示例

■ $R(A,B,C,D,E)$

- $R1(A,D), R2(A,B), R3(B,E), R4(C,D,E), R5(A,E)$

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

■ 判断 R 分解为 $p = \{R1, R2, R3, R4, R5\}$ 是否是无损连接的分解

2) Chase示例

1、构造初始表格

R1(A,D), R2(A,B), R3(B,E), R4(C,D,E), R5(AE)

	A	B	C	D	E
AD	a1	b12	b13	a4	b15
AB	a1	a2	b23	b24	b25
BE	b31	a2	b33	b34	a5
CDE	b41	b42	a3	a4	a5
AE	a1	b52	b53	b54	a5

2、处理表格

A→C: 将b23, b53改为b13

B→C: 将b33改为b13

C→D: 将b24, b34, b54改为a4

DE→C: 将第3行和第5行的C改为a3

CE→A: 将第3行和第4行的A改为a1

2) Chase示例

1、构造初始表格

R1(A,D), R2(A,B), R3(B,E), R4(C,D,E), R5(AE)

	A	B	C	D	E
AD	a1	b12	b13	a4	b15
AB	a1	a2	b13	a4	b25
BE	a1	a2	a3	a4	a5
CDE	a1	b42	a3	a4	a5
AE	a1	b52	a3	a4	a5

2、处理表格

$A \rightarrow C$: 将b23, b53改为b13

$B \rightarrow C$: 将b33改为b13

$C \rightarrow D$: 将b24, b34, b54改为a4

$DE \rightarrow C$: 将第3行和第5行的C改为a3

$CE \rightarrow A$: 将第3行和第4行的A改为a1

因此是无损
连接的分解

3) 方法2

- 当**R**分解为两个关系模式**R1**和**R2**时，有一种简便的方法可以测试无损连接性

- $p = \{R1, R2\}$

- p 是无损连接的分解当且仅当下面之一满足

- ◆ $(R1 \cap R2) \rightarrow (R1 - R2)$

- ◆ $(R1 \cap R2) \rightarrow (R2 - R1)$

- ◆ 其中**R1** \cap **R2**指模式的交，返回公共属性

- ◆ **R2** - **R1**表示模式的差集，返回属于**R2**但不属于**R1**的属性集

- 例 **R(A,B,C), F={A→B}**

- $\rho1 = \{R1(A,B), R2(B,C)\}$, $\rho2 = \{R1(A,B), R2(A,C)\}$

- $\rho2$ 是无损连接, $\rho1$ 不是

4、保持函数依赖

- 关系模式**R**的**FD**集在分解后仍在数据库模式中保持不变
 - 给定**R**和**R**上的一个**FD**集**F**, $\rho = \{R_1, R_2, \dots, R_k\}$ 的分解应使**F**被**R_i**上的函数依赖逻辑蕴含
- 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 $\pi_Z(F)$ 表示, 定义为: $\pi_Z(F) = \{X \rightarrow Y \mid X \rightarrow Y \in F^+ \wedge XY \subseteq Z\}$ 。对于**R(U)**上的一个分解 $\rho = \{R_1, R_2, \dots, R_k\}$, 若满足下面条件, 则称分解 ρ 保持函数依赖集**F**:

$$\left(\bigcup_{i=1}^k \pi_{R_i}(F) \right)^+ = F^+$$

(1) 例子

- $R(\text{city}, \text{street}, \text{zip}), F = \{(\text{city}, \text{street}) \rightarrow \text{zip}, \text{zip} \rightarrow \text{city}\}$
- 分解为 $\rho = \{R1(\text{street}, \text{zip}), R2(\text{city}, \text{zip})\}$
- 是否无损连接?
 - $R1 \cap R2 = \{\text{zip}\}, R2 - R1 = \{\text{city}\}, \text{zip} \rightarrow \text{city}$
 - 无损连接
- 是否保持函数依赖?
 - $\pi_{R1}(F) = \{\text{按自反律推出的平凡FD}\}$
 - $\pi_{R2}(F) = \{\text{zip} \rightarrow \text{city}, \text{以及按自反律推出的平凡FD}\}$
 - $\pi_{R1}(F) \cup \pi_{R2}(F) = \{\text{zip} \rightarrow \text{city}\}^+ \neq F^+$
 - 不保持函数依赖

(2) 不保持函数依赖带来的问题

- $R(\text{city}, \text{street}, \text{zip}), F = \{(\text{city}, \text{street}) \rightarrow \text{zip}, \text{zip} \rightarrow \text{city}\}$
- 分解为 $p = \{R1(\text{street}, \text{zip}), R2(\text{city}, \text{zip})\}$
- 在 $R1$ 中插入 $(\text{'a'}, \text{'100081'})$ 和 $(\text{'a'}, \text{'100082'})$
- $R2$ 中插入 $(\text{'Beijing'}, \text{'100081'})$ 和 $(\text{'Beijing'}, \text{'100082'})$
- $R1 \bowtie R2$: 得到

City	Street	Zip
Beijing	a	100081
Beijing	a	100082

- 违反了 $(\text{city}, \text{street}) \rightarrow \text{zip}$, 因为它被丢失了, 语义完整性被破坏

模式分解小结

■ 三种准则

● 无损连接

- ◆ 若R分解为n(n>2)个关系模式，使用Chase方法判断是否无损连接
- ◆ 若R分解为R1和R2，使用 $(R1 \cap R2) \rightarrow (R1 - R2)$ 或 $(R1 \cap R2) \rightarrow (R2 - R1)$ 判断

● 保持函数依赖

$$\left(\bigcup_{i=1}^k \pi_{R_i}(F) \right)^+ = F^+$$

● 既无损连接，又保持函数依赖

四、关系模式的范式

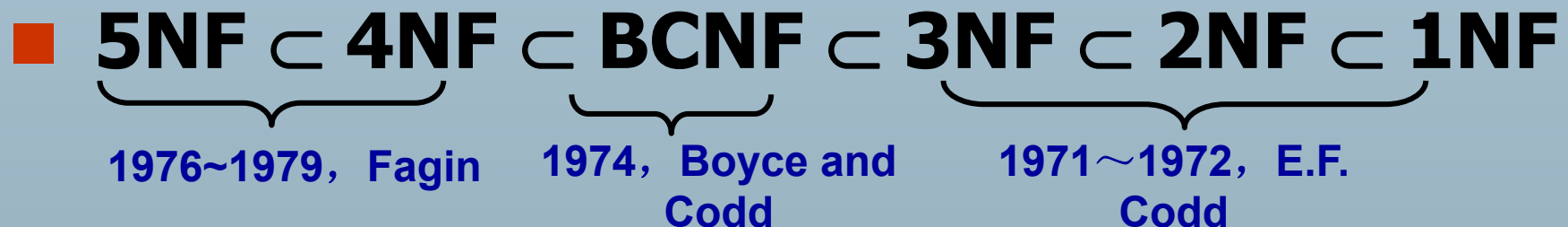
- 范式的概念
- 函数依赖图
- 1NF
- 2NF
- 3NF
- BCNF
- 4NF
- 5NF



模式分解到何时才能结束?

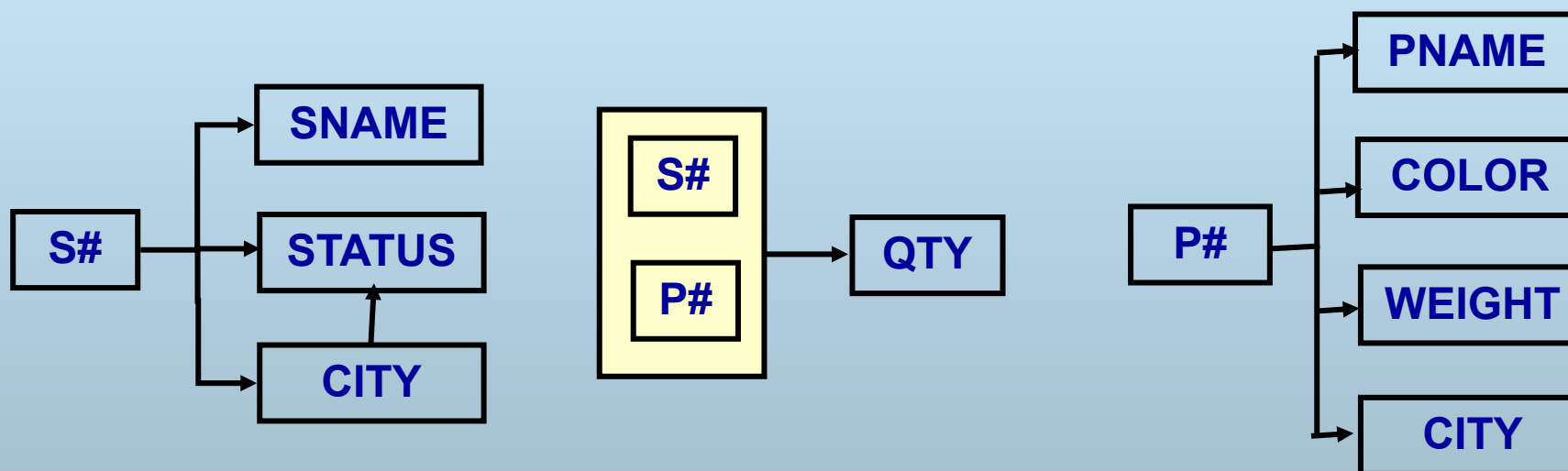
1、范式的概念

- 范式：满足特定要求的模式
 - 不同级别的范式要求各不相同
 - 范式可以作为衡量一个关系模式好坏的标准
 - 若关系模式R满足范式 xNF ，记作 $R \in xNF$
- 规范化：将低一级范式的关系模式通过模式分解转换为高一级范式的关系模式集合的过程



2、函数依赖图

- **R**是关系模式，**F**是**R**的一个**FD**集，**F**可用函数依赖图表达



箭头表示函数决定关系，每个候选码必定有箭头指出

3、1NF

- 对于关系模式R的任一实例，其元组的每一个属性值都只含有一个值，则 $R \in 1NF$
 - 1NF是关系的基本要求
 - R不满足1NF会带来更新时的二义性
 - 若R中加入“成绩”属性，则{学号,课程}→成绩难以表达

学号	课程
01	数据库
02	{C++, 数据库}

4、2NF

- （假定R只有一个候选码/主码）当且仅当R属于1NF，且R的每一个非主属性都完全函数依赖于主码时， $R \in 2NF$
 - 完全函数依赖：对于函数依赖 $W \rightarrow A$ ，若不存在 $X \subset W$ ，并且 $X \rightarrow A$ 成立，则称 $W \rightarrow A$ 为完全函数依赖，否则为局部函数依赖
 - 主属性：包含在候选码中的属性
 - 非主属性：不包含在任何候选码中的属性

(1) 2NF含义

- $R(A,B,C,D,E)$, $\{A,B\}$ 为主码, 则有
- $AB \rightarrow C$, $AB \rightarrow D$, $AB \rightarrow E$
- 但C、D、E都不局部函数依赖于AB
- 即 $A \rightarrow C$ 、 $B \rightarrow C$ 、 $A \rightarrow D$ 、 $B \rightarrow D$ 、 $A \rightarrow E$ 、 $B \rightarrow E$ 中任何一个均不成立

(2) 2NF例子

■ 供应关系

- $R(S\#, P\#, city, status, Price, QTY)$
- $F = \{S\# \rightarrow city, S\# \rightarrow status, P\# \rightarrow Price, city \rightarrow status, \{S\#, P\# \} \rightarrow QTY\}$
- 所以主码为 $\{S\#, P\#\}$
- 但 $city$ 和 $Price$ 都局部函数依赖于主码
- 所以 $R \notin 2NF$

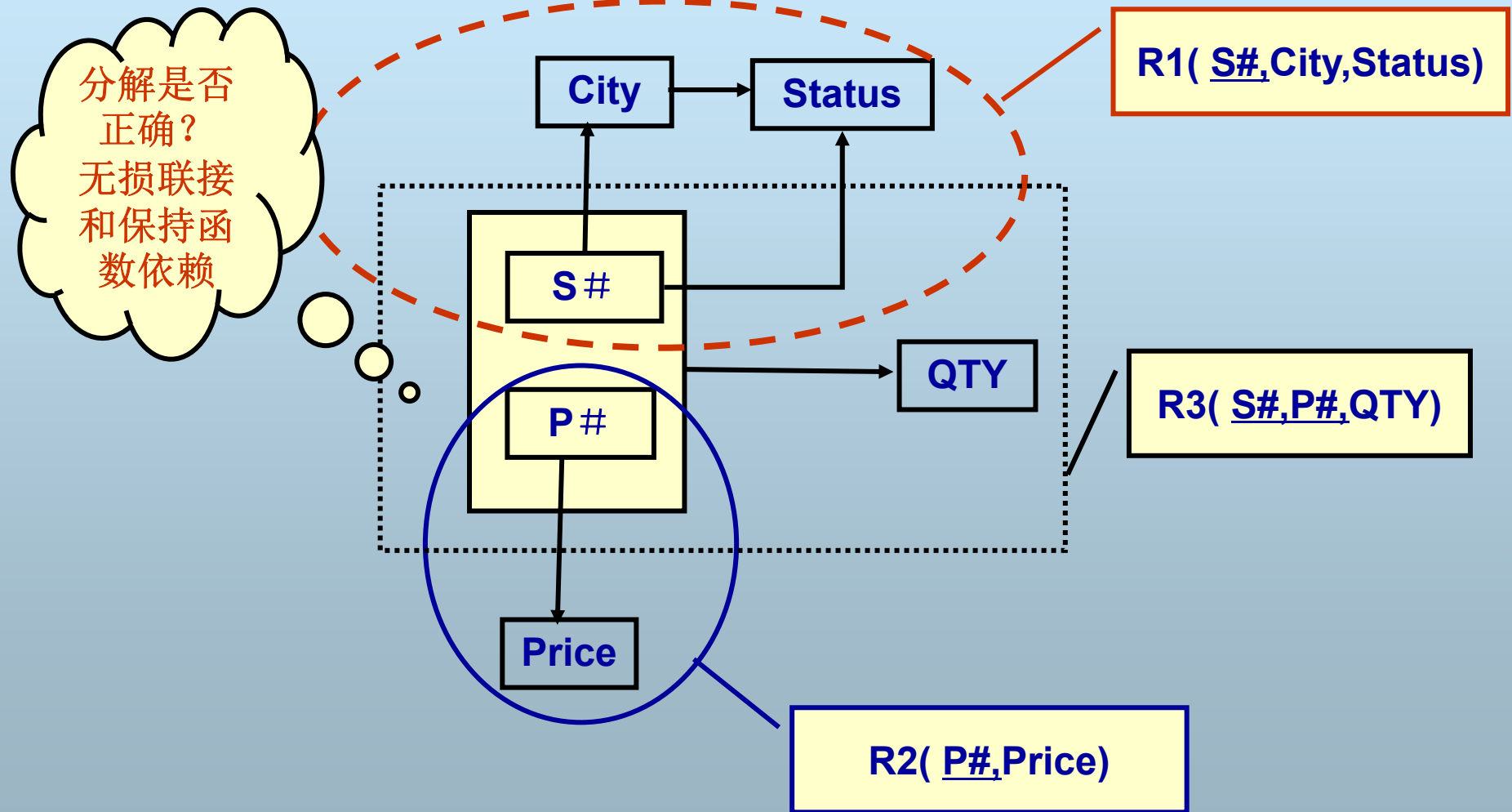
(3) 不满足2NF带来的问题

■ R(S #, P #, city, status, Price, QTY)

- **插入异常**：没有供应零件的供应商无法插入
- **删除异常**：删除供应商的供货信息同时删除了供应商的其它信息
- **更新异常**：供应商的city修改时必须修改多个元组
- **数据冗余**：同一供应商的city被重复存储

(3) 模式分解以满足2NF

■ R(S #, P #, City, Status, Price, QTY)



5、3NF

- (假定R只有一个候选码，且该候选码为主码)
当且仅当R属于2NF，且R的每一个非主属性都不传递依赖于主码时， $R \in 3NF$
- 传递依赖：若 $Y \rightarrow X$ ， $X \rightarrow A$ ，并且 $X \not\rightarrow Y$ ，A不是X的子集，则称A传递依赖于Y

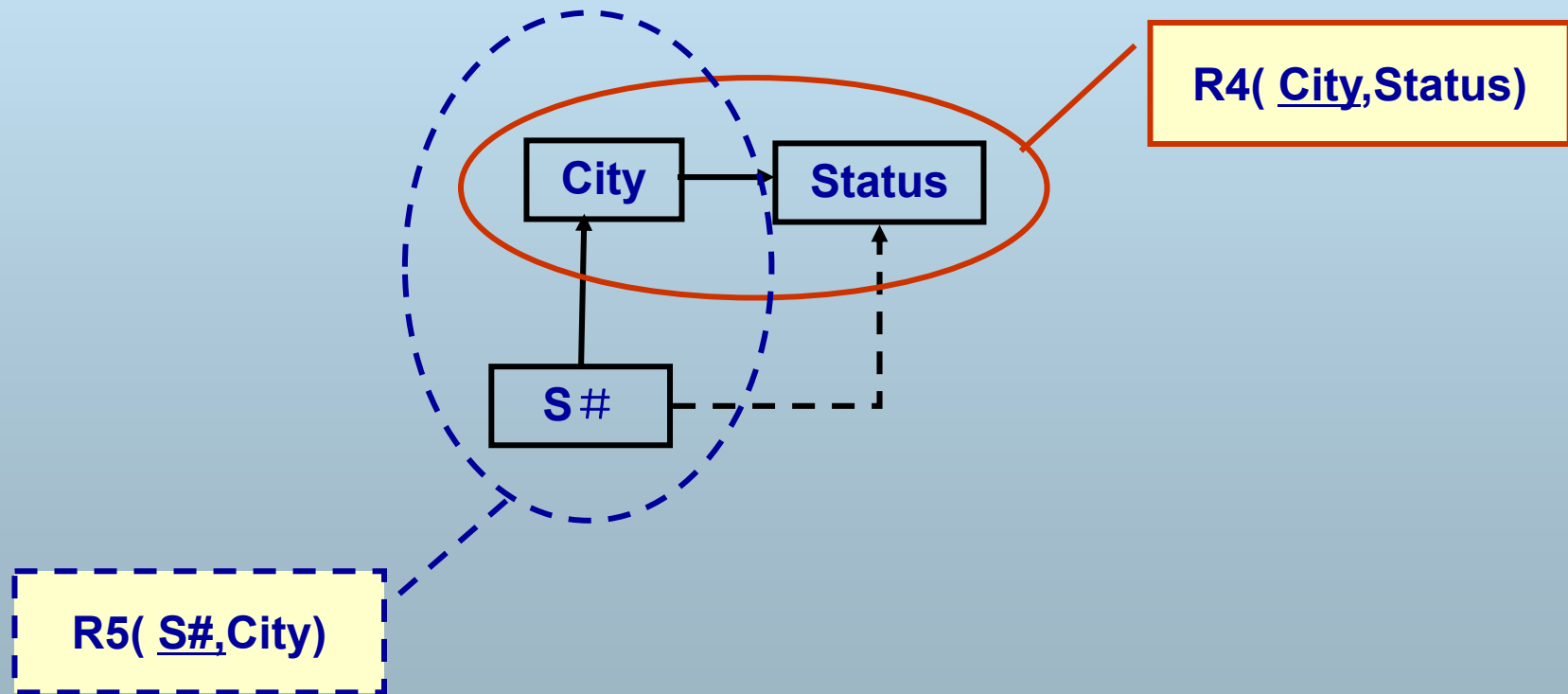
(1) 不满足3NF带来的问题

■ R1(S#,City,Status)

- **插入异常**：不能插入一个具有**status**但没有供应商的**city**，例如**Rome**的**status**为**50**，但除非有一个供应商住在**Rome**否则无法插入
- **删除异常**：删除供应商时会同时删除与该城市相关的**status**信息
- **更新异常**：一个城市中会有多个供应商，因此**status**更新时要更新多个元组
- **数据冗余**：同一城市的**status**冗余存储

(2) 分解2NF到3NF

- **R1(S#,City,Status)**
- 去掉传递依赖



6、BCNF

■ Boyce/Codd范式

■ 2NF和3NF

- 假设了R只有一个候选码，但如果R有多个候选码并且不同的候选码之间还可能相互重叠，会出现什么情况？
- 2NF和3NF只考虑了非主属性到码的函数依赖

■ BCNF扩充了3NF，可以处理R有多个候选码的情形

- 进一步考虑了主属性到码的函数依赖
- 进一步考虑了主属性对非主属性的函数依赖

(1) 多候选码的例子

- 假设供应商的名字是唯一的
- 供应关系 $R(S\#, SNAME, P\#, QTY)$ 存在两个候选码
 - $\{S\#, P\#\}$ 和 $\{SNAME, P\#\}$
 - R 属于3NF, WHY?

$\{SNAME, P\#\} \rightarrow QTY, \{S\#, P\#\} \rightarrow QTY,$
 $S\# \rightarrow SNAME, SNAME \rightarrow S\#$

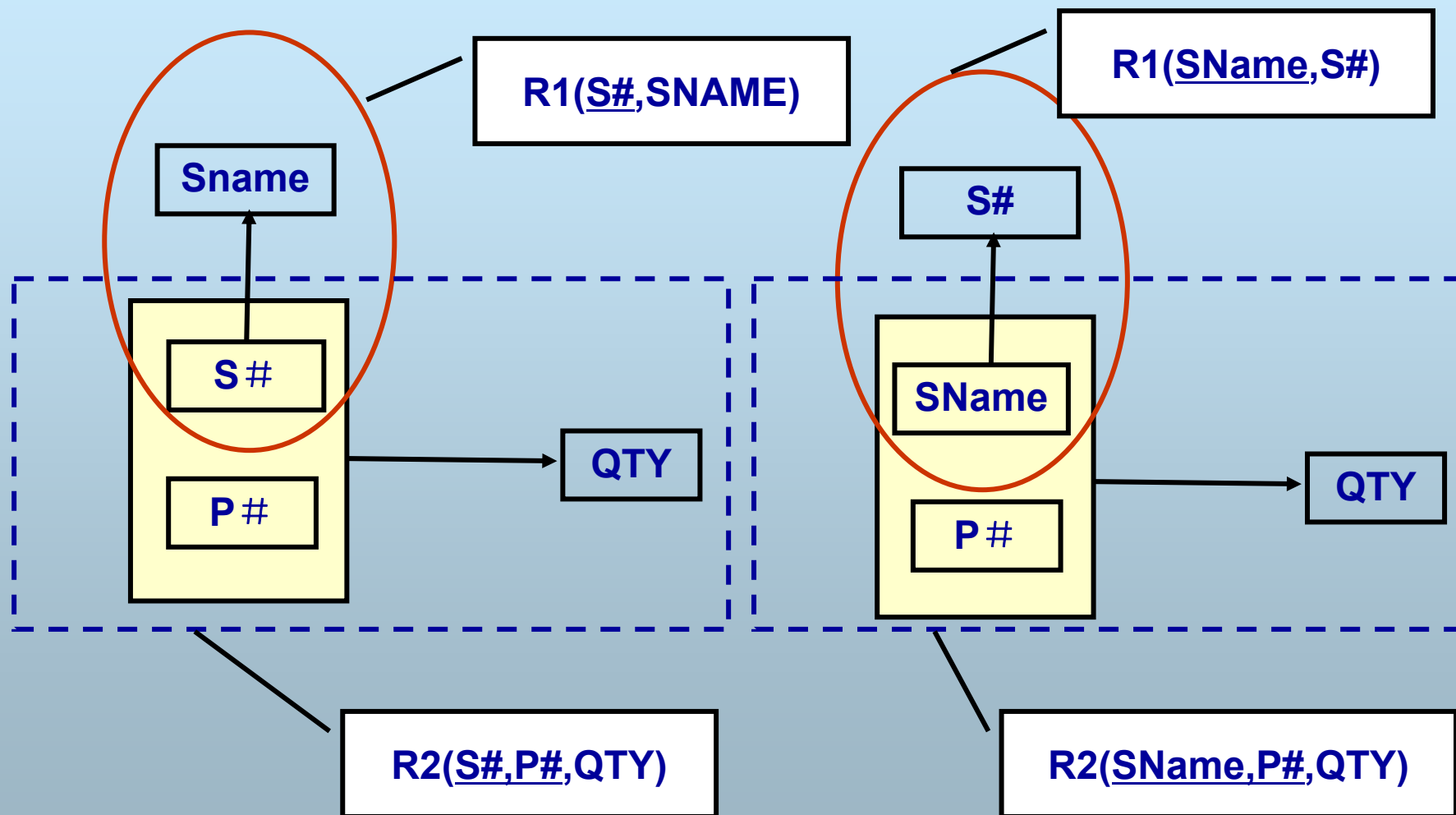
S#	SNAME	P#	QTY
s1	Intel	p1	300
s1	Intel	p2	200
s1	Intel	P3	400
s2	Acer	p1	200

(2) 存在的问题

- **数据冗余：** s1的名字Intel重复存储
- **更新异常：** 修改s1的名字时必须修改多个元组
- **删除异常：** 若s2现在不提供任何零件，则须删除s2的元组，但同时删除了s2的名字
- **插入异常：** 没有提供零件的供应商无法插入

S#	SNAME	P#	QTY
s1	Intel	p1	300
s1	Intel	p2	200
s1	Intel	P3	400
s2	Acer	p1	200

(3) 解决方法 (3NF->BCNF)



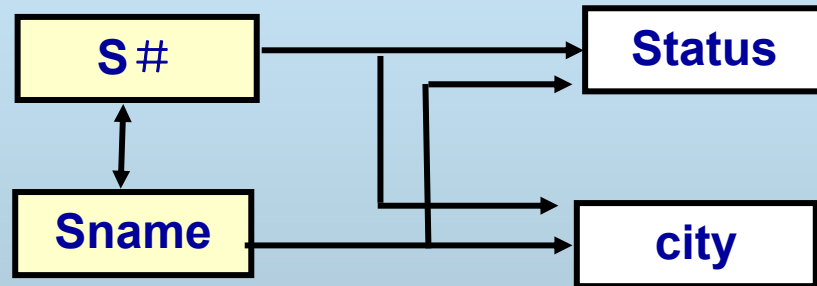
(4) BCNF定义

- (C. J. Date) 如果关系模式R的所有不平凡的、完全的函数依赖的决定因素（左边的属性集）都是候选码，则 $R \in \text{BCNF}$
- 3NF: 不允许非主属性到非码的FD，但允许主属性到其它属性的FD
- BCNF: 不允许主属性、非主属性到非码的FD

C. J. Date, An Introduction to Database Systems

(5) BCNF例子1

- $R(S\#, SNAME, STATUS, CITY)$
- 设Sname唯一



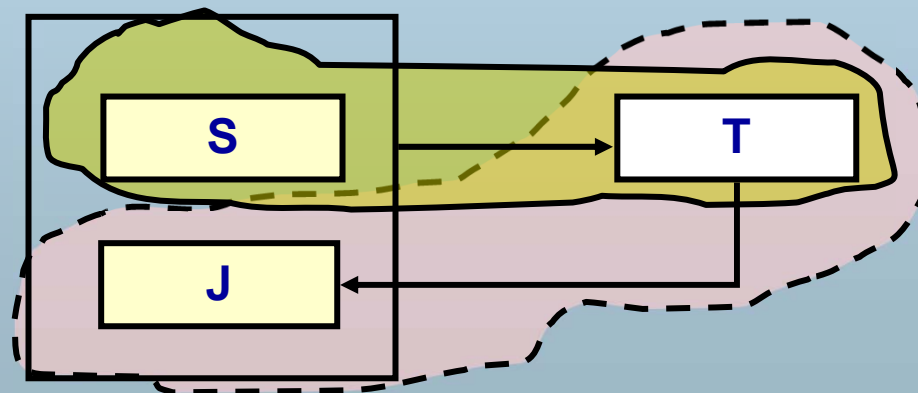
$Sname \rightarrow city,$
 $S\# \rightarrow city,$
 $S\# \rightarrow Sname,$
 $Sname \rightarrow S\#,$
 $Sname \rightarrow Status,$
 $S\# \rightarrow Status$

- BCNF模式的函数依赖图中，箭头都是从候选码中引出，所有不平凡FD的左边都是候选码

(6) BCNF例子2

- $R(S, J, T)$ --- 学号, 课程号, 教师名
- 每个教师只教一门课, 每门课有若干任课教师, 学生选定一门课就对应一个固定的教师
- $T \rightarrow J, \{S, J\} \rightarrow T$
- R属于3NF
- R不属于BCNF

$R1(\underline{S}, T)$
 $R2(\underline{T}, J)$



分解到BCNF不一定能保持函数依赖

五、规范化过程总结

- 对**1NF**模式投影，消除非主属性对码的局部函数依赖，产生**2NF**
- 对**2NF**模式投影，消除非主属性对码的传递函数依赖，产生**3NF**
- 对**3NF**模式投影，消除左边不是候选码的函数依赖，产生**BCNF**

五、规范化过程总结

- 整个讨论过程只采用了两种操作：投影和自然联接
 - 以投影来分解
 - 以自然连接来重构

五、规范化过程总结

- **定理1**：若要求保持函数依赖和无损联接，则总可以达到**3NF**，但不一定满足**BCNF**
- **定理2**：若要求模式分解保持函数依赖，则总可以分解到满足**3NF**，但不一定满足**BCNF**
 - **BCNF**可以达到无损连接，但不一定保持函数依赖

六、模式分解的几个算法

■ 算法1

- 保持函数依赖地分解到**3NF**的算法

■ 算法2

- 无损并且保持函数依赖分解为**3NF**的算法

■ 算法3

- 无损分解为**BCNF**的算法

算法1：保持函数依赖地分解到3NF

1. 求出 $R\langle U, F \rangle$ 的最小函数依赖集（仍记为 F ）
2. 把所有不在 F 中出现的属性组成一个关系模式 R' ，并在 U 中去掉这些属性(剩余属性仍记为 U)
3. 若 F 中存在 $X \rightarrow A$ ，且 $XA=U$ ，则输出 $R(U)$ 和 R' ，算法结束，否则
4. 对 F 按相同的左部分组，将所有 $X \rightarrow A_1, X \rightarrow A_2, \dots, X \rightarrow A_k$ 形式的FD分为一组，并将每组涉及的所有属性作为一个关系模式输出。若某个关系模式 R_i 的属性集是另一个关系模式的属性集的子集，则在结果中去掉 R_i 。设最后得到关系模式 R_1, R_2, \dots, R_k ，则 $p=\{R_1, R_2, \dots, R_k, R'\}$ 一个保持函数依赖的分解，并且满足3NF

例子

- $R(ABCDEF), F=\{A \rightarrow B, AC \rightarrow E\}$
- 求最小FD集 $F=\{A \rightarrow B, AC \rightarrow E\}$
- $R'(DF)$
- 按左部分组: $R1(AB), R2(ACE)$
- $p=\{R'(DF), R1(AB), R2(ACE)\}$

算法2：无损连接且保持函数依赖地分解到3NF

- 首先用算法1求出R的保持函数依赖的3NF分解，设为 $q=\{R_1, R_2, \dots, R_k\}$
- 设X是R的主码，求出 $p=q \cup \{R(X)\}$
- 若X是q中某个 R_i 的子集，则在p中去掉 $R(X)$
- 得到的p就是最终结果

(1) 例子1

- $R(S\#, SN, P, C, S, Z),$
 $F = \{S\# \rightarrow SN, S\# \rightarrow P, S\# \rightarrow C, S\# \rightarrow S, S\# \rightarrow Z, \{P, C, S\} \rightarrow Z, Z \rightarrow P, Z \rightarrow C\}$
- 1. 求出最小FD集: $F = \{S\# \rightarrow SN, S\# \rightarrow P, S\# \rightarrow C, S\# \rightarrow S, \{P, C, S\} \rightarrow Z, Z \rightarrow P, Z \rightarrow C\}$ // $S\# \rightarrow Z$ 冗余
- 2. $q = \{R1(S\#, SN, P, C, S), R2(P, C, S, Z), R3(Z, P, C)\}$
- 3. $R3$ 是 $R2$ 的子集, 所以去掉 $R3$
 $q = \{R1(S\#, SN, P, C, S), R2(P, C, S, Z)\}$
- 4. R 的主码为 $S\#$, 于是
 $p = q \cup \{R(X)\} = \{R1(S\#, SN, P, C, S), R2(P, C, S, Z), R(S\#)\}$
- 5. 因为 $\{S\# \}$ 是 $R1$ 的子集, 所以从 p 中去掉 $R(S\#)$
- 6. $p = \{R1(S\#, SN, P, C, S), R2(P, C, S, Z)\}$ 即最终结果

(2) 例子2

- $R(S\#, SN, P, C, S, Z)$,
 $F = \{S\# \rightarrow SN, S\# \rightarrow P, S\# \rightarrow C, Z \rightarrow S, Z \rightarrow C\}$
- 1. 求出最小FD集: $F = \{S\# \rightarrow SN, S\# \rightarrow P, S\# \rightarrow C, Z \rightarrow S, Z \rightarrow C\}$
- 2. $q = \{R1(S\#, SN, P, C), R2(Z, S, C)\}$
- 3. R的主码为 $\{S\#, Z\}$, 于是
 $p = q \cup \{R(X)\} = \{R1(S\#, SN, P, C), R2(Z, S, C), R(S\#, Z)\}$
- 4. $p = \{R1(S\#, SN, P, C), R2(Z, S, C), R(S\#, Z)\}$ 即最终结果

算法3：无损联接地分解R到BCNF

- 输入： $R\langle U, F \rangle$; 输出： p
- 1. $p := \{R\}$;
- 2. 检查 p 中各关系模式是否都属于BCNF，若是，则算法终止
- 3. 设 p 中 $S(U_s)$ 非BCNF关系模式, 则必存在 $X \rightarrow A$, 其中 X 不是 S 的超码;
 - ① 将 S 分解为 $S1(XA)$ 和 $S2(U_s - A)$, 此分解是无损联接的
// $(\{XA\} \cap \{U_s - A\} = X) \rightarrow (A = \{XA\} - \{U_s - A\})$
 - ② $p := \{p - S\} \cup \{S1, S2\}$; // 用 $S1$ 和 $S2$ 替换 p 中的 S
 - ③ 转到第2步;
- 4. 由于 U 的属性有限，因此有限次循环后算法终止

例子

- $R(S\#, C\#, G, TN, D),$
 $F = \{\{S\#, C\#\} \rightarrow G, C\# \rightarrow TN, TN \rightarrow D\}$
- $\rho := \{R\};$
- $TN \rightarrow D$ 不满足BCNF定义, 分解R
 $\rho := \{R1(S\#, C\#, G, TN), R2(TN, D)\}$
- R1中 $C\# \rightarrow TN$ 不满足BCNF, 分解R1为R3和R4
 $\rho := \{R3(S\#, C\#, G), R4(C\#, TN), R2(TN, D)\}$
- ρ 中各模式均满足BCNF, 结束

例子（续）

- $R(S\#,C\#,G,TN,D),$
 $F=\{\{S\#,C\#\} \rightarrow G, C\# \rightarrow TN, TN \rightarrow D\}$
- 如果先选择处理 $C\# \rightarrow TN$?
 - $C\# \rightarrow TN$ 不满足BCNF定义，分解R
 $\rho := \{R1(\underline{S\#}, \underline{C\#}, G, \underline{D}), R2(\underline{C\#}, TN)\}$
 - R1中 $\{S\#,C\#\} \rightarrow G$ 不满足BCNF，分解R1为R3和R4
 $\rho := \{R3(\underline{S\#}, \underline{C\#}, G), R4(\underline{S\#}, \underline{C\#}, \underline{D}), R2(\underline{C\#}, TN)\}$
 - ρ 中各模式均满足BCNF，结束

结论：无损分解到BCNF的结果不唯一！

本章小结

- 模式设计理论是数据库逻辑设计的理论基础，目的是根据初始的数据库模式构造出合适的数据库模式
- 函数依赖
- 模式分解
 - 无损联接
 - 保持函数依赖
- 规范化理论
 - 1NF、2NF、3NF、BCNF
- 模式分解的算法