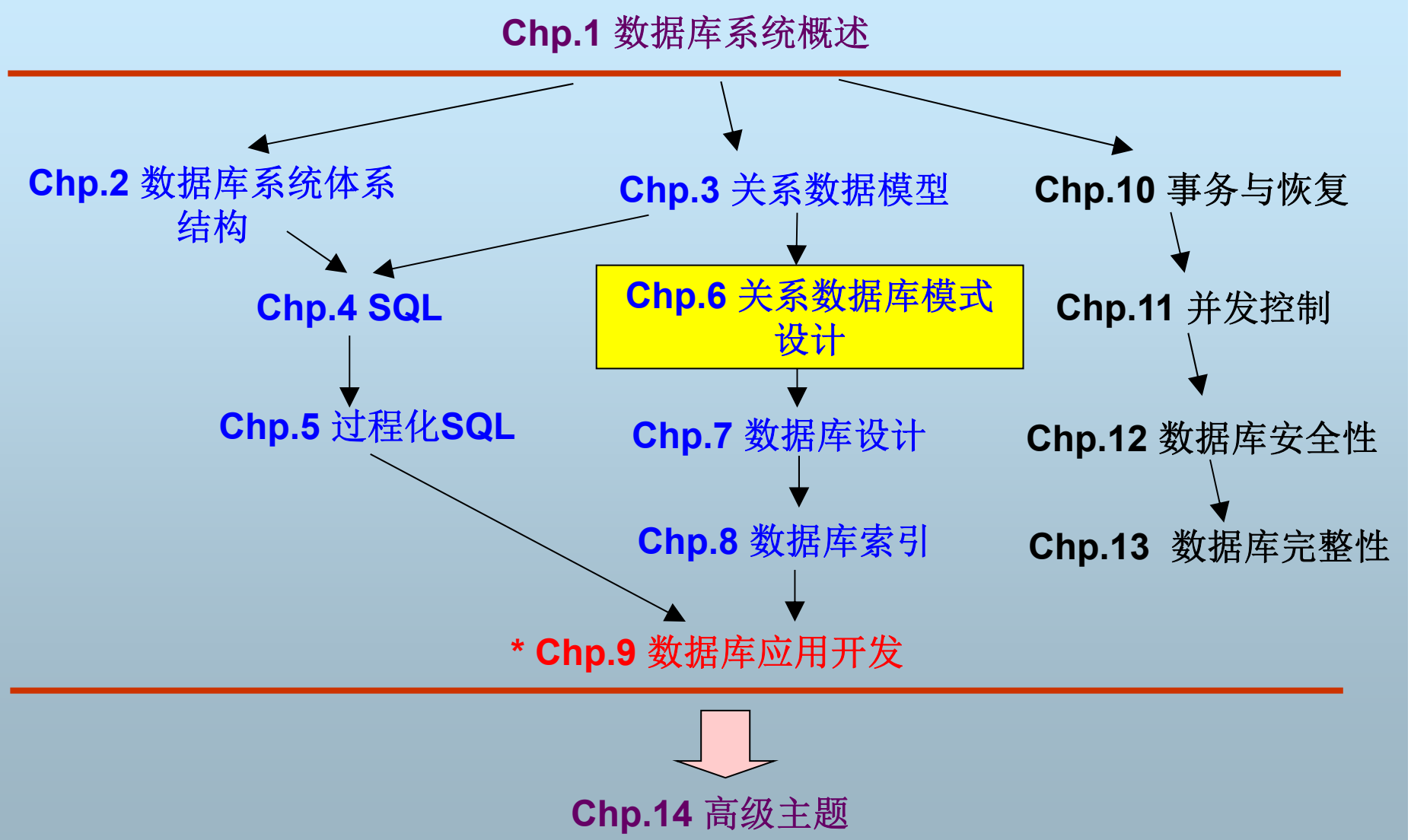
第6章 关系数据库模式设计

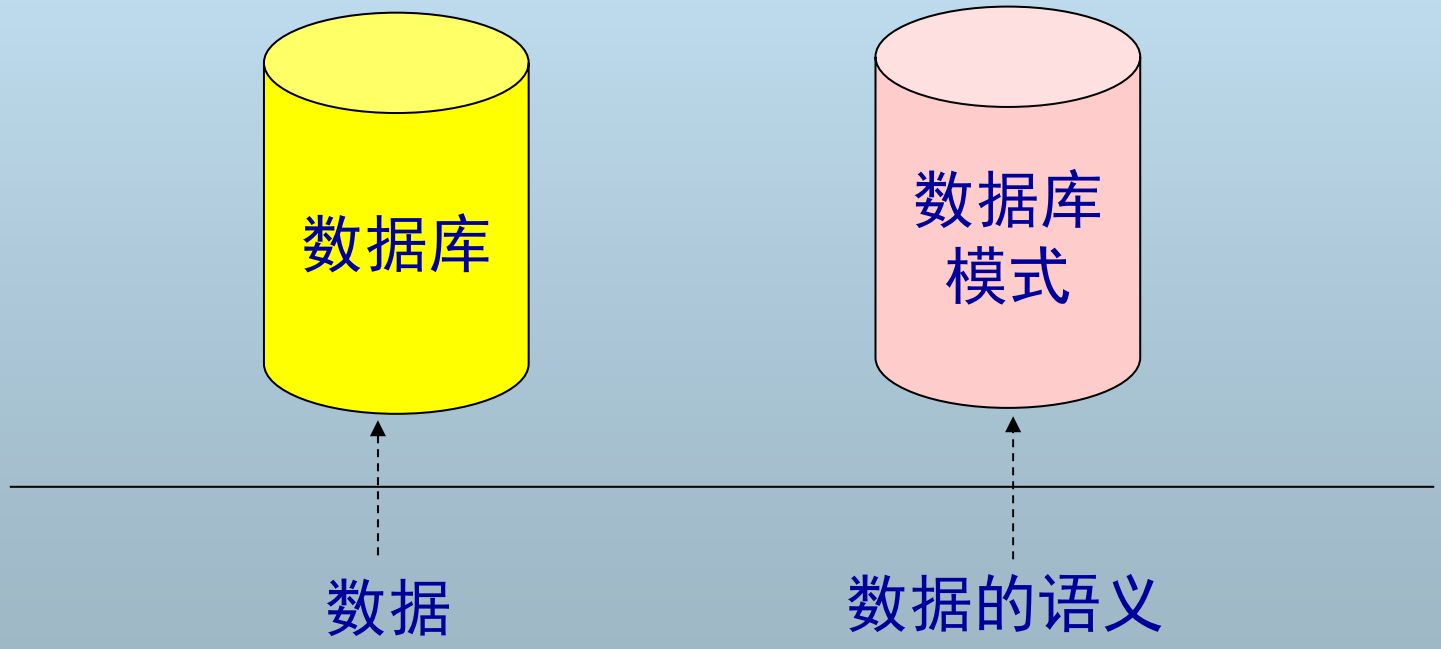
金培权（jpq@ustc.edu.cn）

课程知识结构


Chp.1 数据库系统概述
Chp.2 数据库系统体系结构
Chp.3 关系数据模型
Chp.10 事务与恢复
Chp.4 SQL
Chp.6 关系数据库模式设计
Chp.11 并发控制
Chp.5 过程化SQL
Chp.7 数据库设计
Chp.12 数据库安全性
Chp.8 数据库索引
Chp.13 数据库完整性
* Chp.9 数据库应用开发
Chp.14 高级主题

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	C#	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	C#	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	C#	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	C#	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	C#	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)
  - 方法2
    - R1(Tname, Addr, Tel, C#)
    - R2(C#, Cname)
  - 方法3
    - R1(Tname, Addr, Tel)
    - R2(Tname ,C#, Cname)

授课信息丢失了

R1中问题依然存在

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？

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(A1,A2,...,An)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t1、t2，由t1[X]=t2[X]可导致t1[Y]=t2[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→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#} →{Sname} （单个属性可去掉括号，简写成 S# →Sname ）
  - SC关系模式中， {S#,C#} →{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#} →{Tname}
    - 若一门课可有多个教师任教，则{C#} →{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→Y，则称F逻辑蕴含X→Y，记作F⊨X→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（Reflexity）：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→BC, B→E, CD→EF}
- AD→F对于函数依赖集F是否成立？
  - A→BC（已知）
  - A→C （分解律）
  - AD→CD（增广律）
  - CD→EF（已知）
  - AD→EF（传递律）
  - AD→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→U ∈ 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→U成立，则X是R的一个候选码
- R(Tname, Addr, C#, Cname)
  - F={Tname→Addr, C#→Cname, C#→Tname}
  - C#→{Tname,Addr,C#,Cname}
  - 所以C#是候选码，若C#→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→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→A中所有A的集合，称为属性集X关于F的闭包，记做X^+
- X→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→B, B→C, B→D, A→D }
  - A^+=ABCD
  - B^+=BCD
  - C^+=C
  - D^+=D
- 不用计算F^+，就可知A→CD ∈ 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#→Sname, S#→age, S#→sex}是最小函数依赖集**
  - **F2={S#→{S#,Sname}, S#→age, S#→sex}不是最小函数依赖集【右边不是单属性】**
  - **F3={S#→Sname, {S#,Sname}→age, S#→sex}不是最小函数依赖集【左部可约】**
  - **F4={S#→S#, S#→Sname, S#→age, S#→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→BC, B→C, A→B, AB→C, AC→D}
  - 将右边写出单属性并去除重复FD（分解律）
    - F={A→B, A→C, B→C, A→B, AB→C, AC→D}
    - F={A→B, A→C, B→C, AB→C, AC→D}
  - 消去左部冗余属性
    - A→C， AC→D可推出A→AC， A→D，因此可去除AC→D中的C
    - A→C，可推出AB→BC可得AB→C ，所以AB →C中的B是冗余属性
    - F={A→B, A→C, B→C, A→C, A→D}
={A→B, A→C, B→C, A→D}
  - 消去冗余函数依赖
    - A→C冗余，因为可由A→B， B→C推出
    - F={A→B, B→C, A→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R1(U1)，R2(U2)，…，Rk(Uk)，其中U＝U1 ∪ U2 … ∪ Uk，设ρ＝{R1, R2,…, Rk}，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→B, C→B}, 则分解为R1(A,B), R2(A,C)丢失了C→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

S#	Status
S3	30
S5	30

Status	City
30	Paris
30	Athens

∞

S#	Status	City
S3	30	Paris
S3	30	Athens
S5	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

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ρ={R1,R2,...,Rk}，F是R上的一个FD集，若对R中满足F的每个关系r，都有：$r=\pi_{R1}(r) \bowtie \pi_{R2}(r) \bowtie \ldots \bowtie \pi_{Rk}(r)$，则称这个分解p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 $m_\rho(r) = \overset{k}{\underset{i=1}{\infty}} \pi_{Ri}(r)$
- 则对于关系模式R关于F的无损连接条件是r＝m_ρ(r)

（2）概念

R

S#	Status	City
S3	30	Paris
S5	30	Athens

$r \neq m_p(r)$

所以不是无损连接

R1

S#	Status
S3	30
S5	30

R2

Status	City
30	Paris
30	Athens

(1)Select * From R

(2)Select * From R1,R2
where
R1.Status=R2.Status

返回结果不一致

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

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

（3）无损连接的测试

- 方法1：Chase
  - 输入：关系模式R(A1,A2,...,An)，R上的函数依赖集F，R的一个分解p={R1,...,Rk}
  - 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≤i ≤ k)，每列对应一个属性A_j（ 1≤j≤ 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→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$a_1a_2...a_n$，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→C, B→C, C→D, DE→C, CE→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

因此是无损连接的分解

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

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 ∩ R2）→（R1－R2）
    - （R1 ∩ R2）→（R2－R1）
    - 其中R1 ∩ 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}
  - ρ1={R1(A,B),R2(B,C)}, ρ2={R1(A,B),R2(A,C)}
  - ρ2是无损连接， ρ1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， ρ＝{R1,R2,..., Rk}的分解应使F被Ri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π_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ρ ＝{R1,R2,..., Rk},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(city, street, zip), F={(city,street)→zip, zip→city}
- 分解为ρ={R1(street,zip),R2(city,zip)}
- 是否无损连接？
  - R1∩R2={zip}, R2－R1={city}, zip→city
  - 无损连接
- 是否保持函数依赖？
  - $\pi_{R1}(F)$={按自反律推出的平凡FD}
  - $\pi_{R2}(F)$={zip→city，以及按自反律推出的平凡FD}
  - $\pi_{R1}(F) \cup \pi_{R2}(F)=\{zip\rightarrow city\}^{+} \neq F^{+}$
  - 不保持函数依赖

（2）不保持函数依赖带来的问题

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

City	Street	Zip
Beijing	a	100081
Beijing	a	100082

- 违反了(city,street)→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 ∩ R2) → (R1－R2)或(R1 ∩ R2) → (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

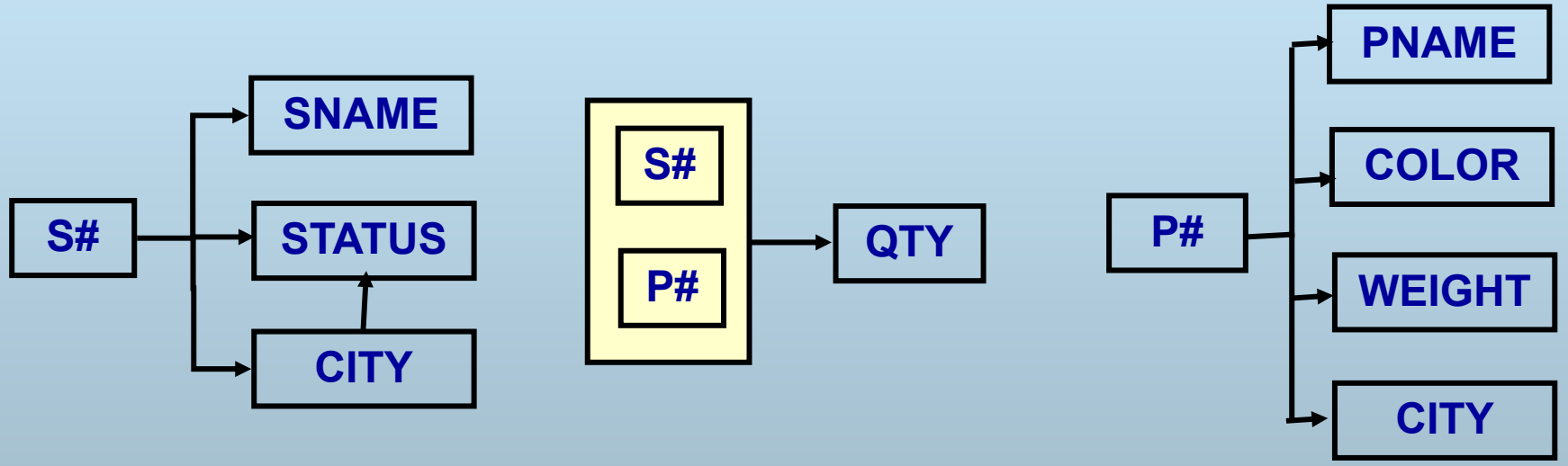
1、范式的概念

- 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$\in x$NF
- 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
- $5NF \subset 4NF \subset BCNF \subset 3NF \subset 2NF \subset 1NF$

1976~1979，Fagin　　1974，Boyce and Codd　　1971～1972，E.F. Codd

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∈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∈2NF
  - 完全函数依赖：对于函数依赖W→A，若不存在X⊂W，并且X→A成立，则称W→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→C， AB→D， AB→E
- 但C、D、E都不局部函数依赖于AB
- 即A→C、 B→C、 A→D、 B→D、 A→E、 B→E中任何一个均不成立

（2）2NF例子

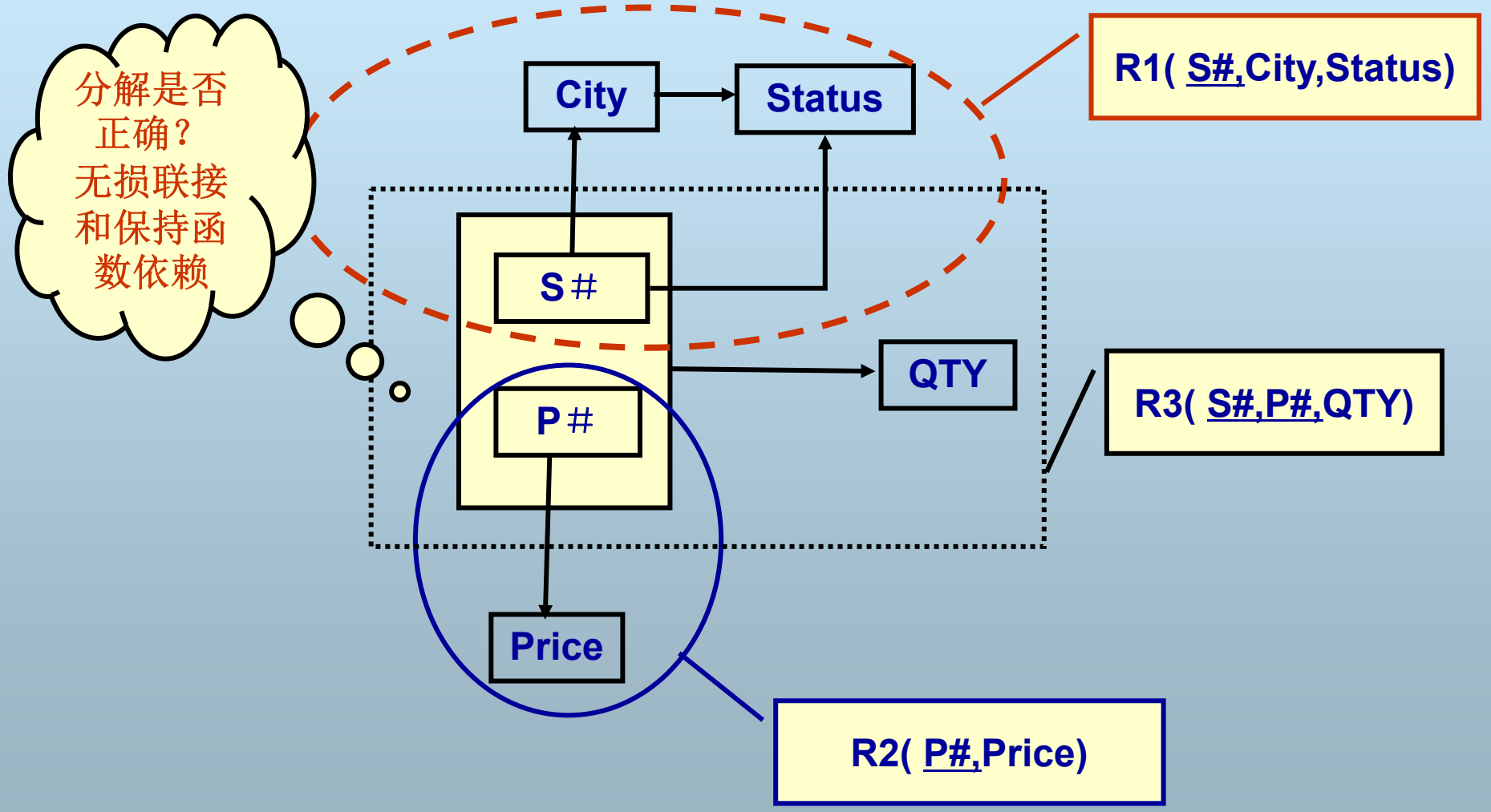
- 供应关系
  - R(S＃, P＃, city, status,Price, QTY)
  - F={S#→city, S#→status , P#→Price, city → status,{S#,P#} →QTY }
  - 所以主码为{S＃,P＃}
  - 但city和Price都局部函数依赖于主码
  - 所以R∉ 2NF

（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

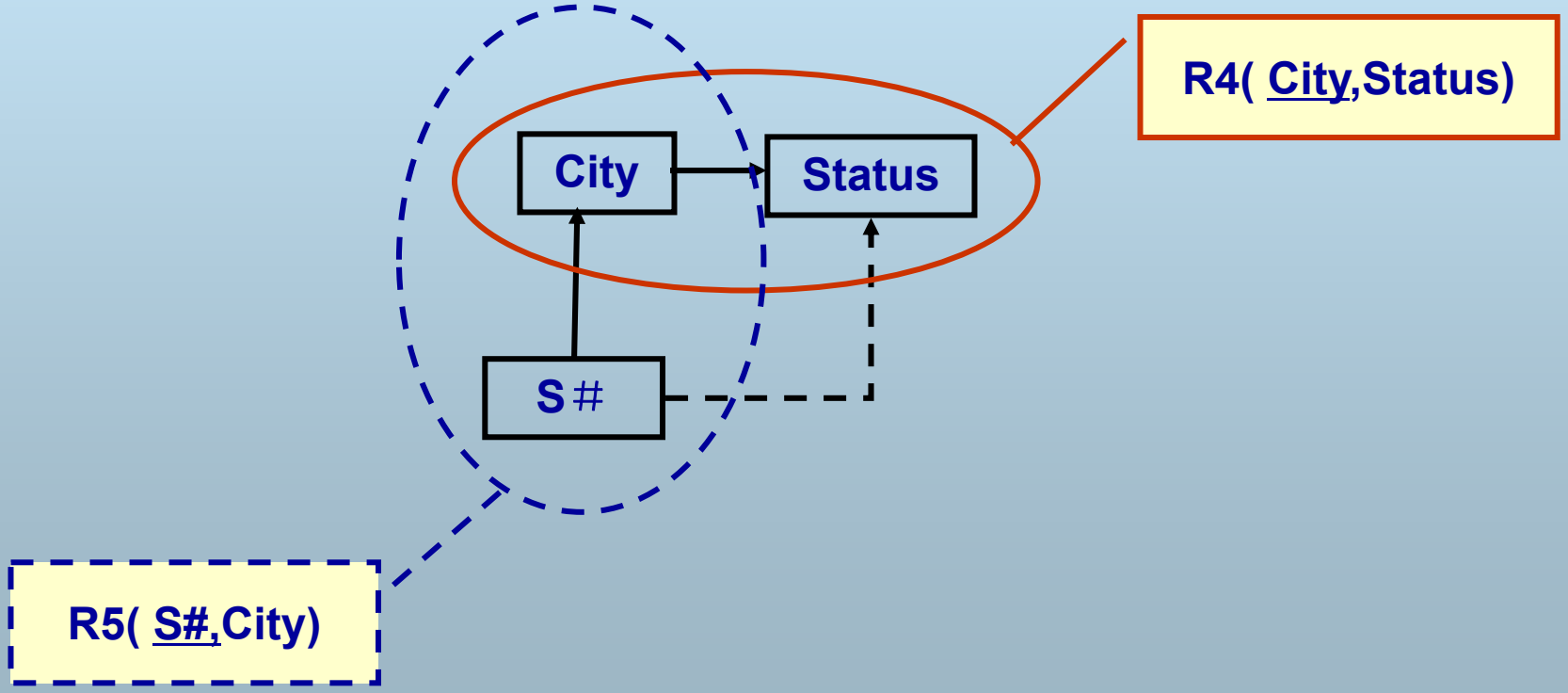
- (假定R只有一个候选码，且该候选码为主码)当且仅当R属于2NF，且R的每一个非主属性都不传递依赖于主码时，R∈3NF
  - 传递依赖：若Y→X，X→A，并且X↛Y，A不是X的子集，则称A传递依赖于Y

（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#} →QTY,{S#,P#} →QTY,
S# →SNAME, SNAME →S#

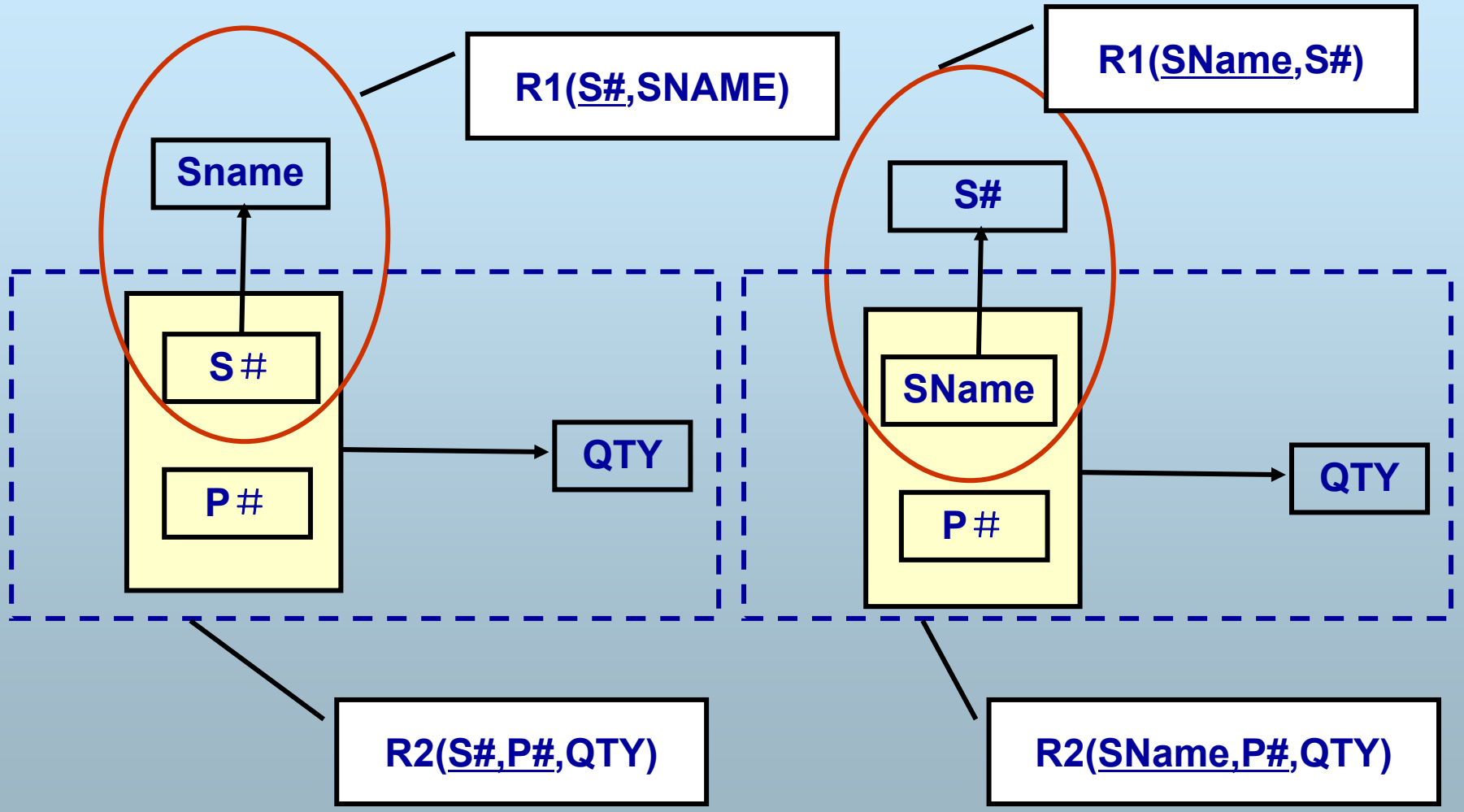
S#	SNAME	P#	QTY
s1	Intel	p1	300
s1	Intel	p2	200
s1	Intel	P3	400
s2	Acer	p1	200

（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）



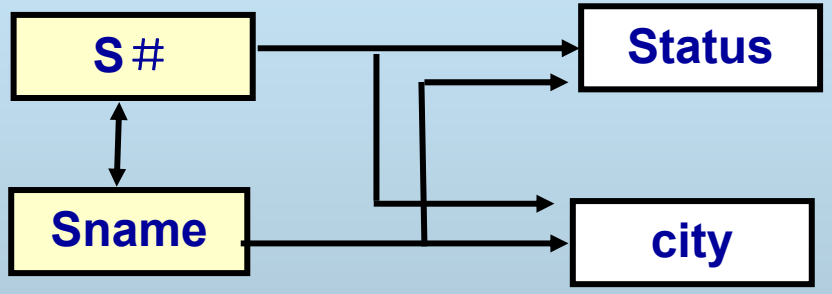
（4）BCNF定义

- （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∈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



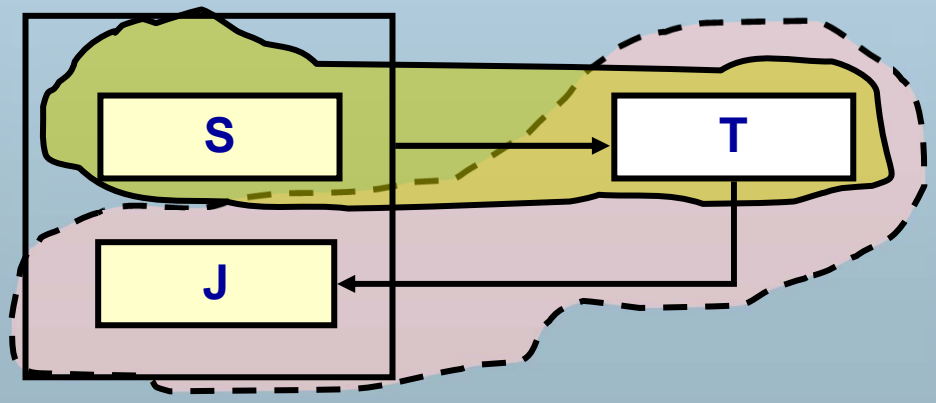
Sname →city,
S# →city,
S# →Sname,
Sname →S#,
Sname→Status,
S#→Status

- 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→J，{S,J} →T
- R属于3NF
- R不属于BCNF

R1(S, T)
R2(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<U,F>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 →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 →A1, X →A2,..., X →Ak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Ri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Ri。设最后得到关系模式R1，R2，...，Rk，则p={R1,R2,...,Rk,R′}一个保持函数依赖的分解，并且满足3NF

例子

- R(ABCDEF), F={A→B,AC→E}
- 求最小FD集F={A→B,AC→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={R1,R2,...,Rk}
- 设X是R的主码，求出p=q ∪ {R(X)}
- 若X是q中某个Ri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#→SN,S#→P,S#→C,S#→S ,S#→Z,{P,C,S}→Z,
  Z→P,Z→C}

1. 求出最小FD集：F={S# →SN, S# →P,S# →C, S#→S,
   {P,C,S} →Z, Z →P,Z →C} // S# →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 ∪ {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#→SN,S#→P,S#→C, Z→S,Z→C}

1. 求出最小FD集：F={S#→SN,S#→P,S#→C, Z→S,Z→C}
2. q={R1(S#,SN,P,C), R2(Z,S,C)}
3. R的主码为{S#, Z}，于是
   p=q ∪ {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<U,F>;　　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→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}∩{U_s－A}=X)→(A={XA}－{U_s－A})
   - ② p:={p－S}∪{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#} →G, C#→TN, TN→D}
- ρ:={R};
- TN→D不满足BCNF定义，分解R
  ρ:={R1(S#,C#,G,TN), R2(TN,D)}
- R1中C#→TN不满足BCNF，分解R1为R3和R4
  ρ:={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#} →G, C#→TN, TN→D}
- 如果先选择处理 C#→TN？
  - C#→TN不满足BCNF定义，分解R
    - ρ:={R1(S#,C#,G,D), R2(C#,TN)}
    - R2满足BCNF，R1不满足BCNF
      - 注意：R1的FD为：{S#, C#} →G, C#→D，所以码为{S#,C#}
  - R1中的C# →D不满足BCNF，继续分解为
    - ρ:={R3(S#,C#,G), R4(C#,D), R2(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