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

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

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

- 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
S3	30
S5	30

Status	City
30	Paris
30	Athens

$\bowtie$

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 $\rho=\{R1,R2,\ldots,Rk\}$，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 $m_{\rho}(r) = \overset{k}{\underset{i=1}{\infty}} \pi_{Ri}(r)$
- 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

R1

S＃	Status
S3	30
S5	30

R2

Status	City
30	Paris
30	Athens

$\bowtie$

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

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

$r \neq m_p(r)$

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

（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

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≤$i \le k$)，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

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

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 ∩ 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， $\rho=\{R1,R2,\ldots,Rk\}$ 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： $\pi_Z(F)=\{X\to Y \mid X\to Y\in F^+\wedge XY\subseteq Z\}$。对于R(U)上的一个分解$\rho=\{R1,R2,\ldots,Rk\}$,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？
  - π_{R1}(F)＝{按自反律推出的平凡FD}
  - π_{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

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

  - 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

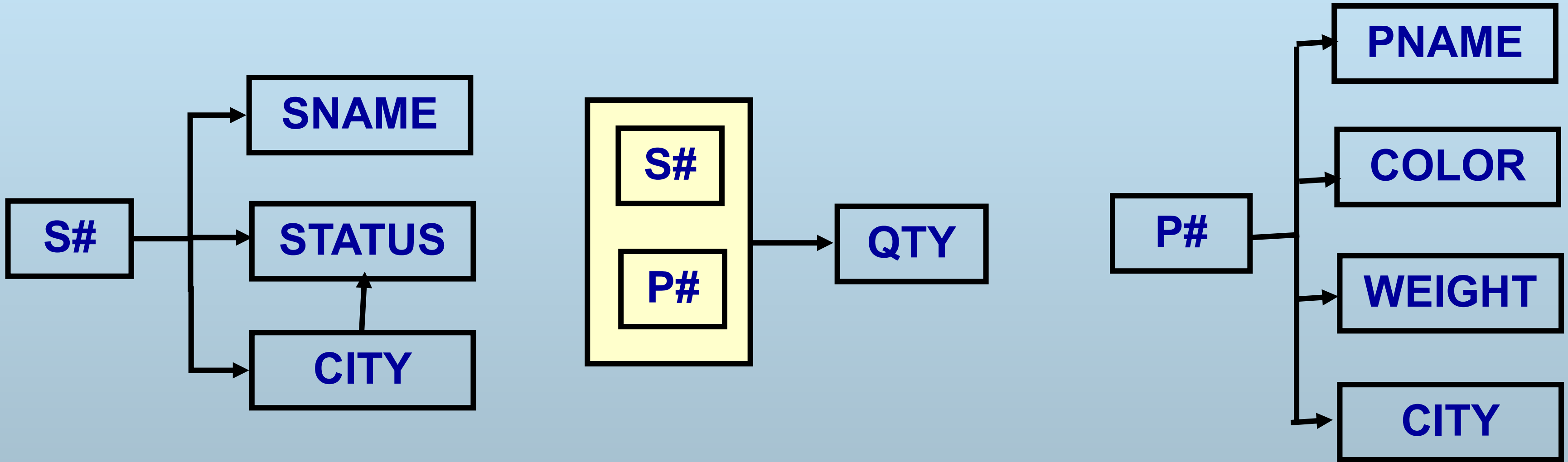
模式分解到何时才能结束？

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 *x*NF，记作 R∈*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
- $\underbrace{\text{5NF} \subset \text{4NF}}_{\text{1976\textasciitilde 1979，Fagin}} \subset \underbrace{\text{BCNF}}_{\text{1974，Boyce and Codd}} \subset \underbrace{\text{3NF} \subset \text{2NF} \subset \text{1NF}}_{\text{1971～1972，E.F. Codd}}$

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 $\in$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

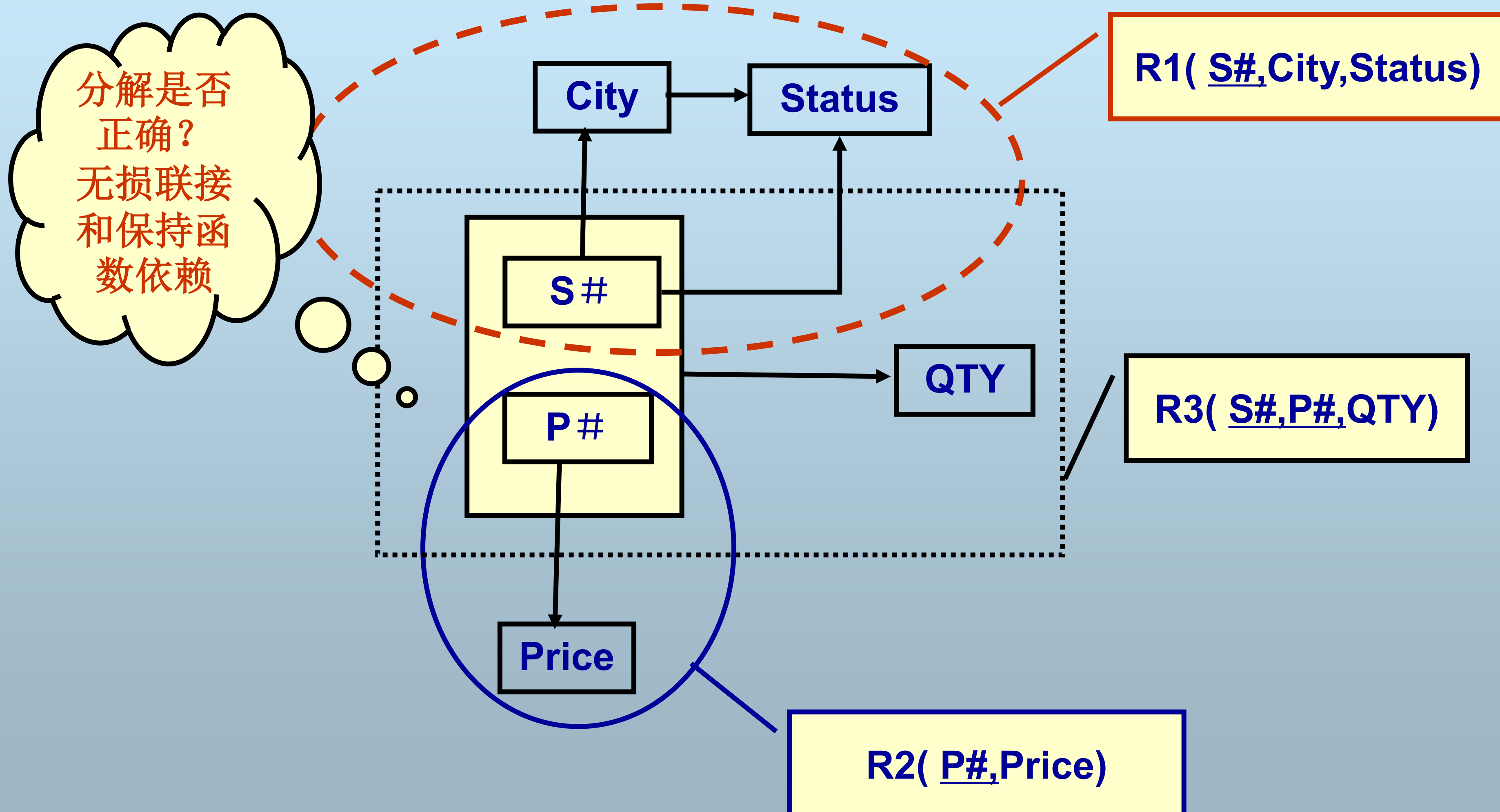
- 供应关系
  - R(S＃, P＃, city, status,Price, QTY)
  - F={S#→city, S#→status , P#→Price, city → status,{S#,P#} →QTY }
  - 所以主码为{S＃,P＃}
  - 但city和Price都局部函数依赖于主码
  - 所以R$\notin$ 2NF

（3）不满足2NF带来的问题

- R(<u>S＃, P＃</u>, 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

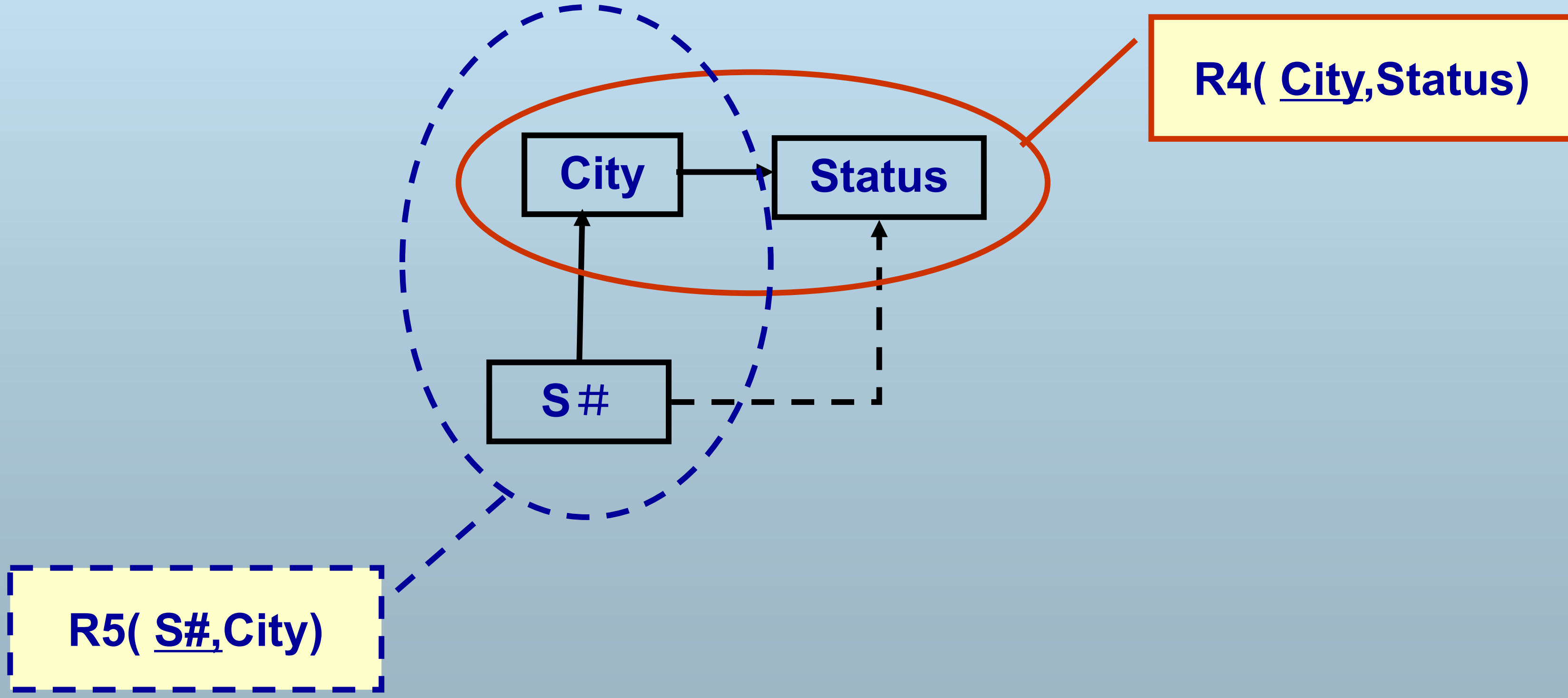
- (假定R只有一个候选码，且该候选码为主码)当且仅当R属于2NF，且R的每一个非主属性都不传递依赖于主码时，R∈3NF
  - 传递依赖：若$Y \rightarrow X$，$X \rightarrow A$，并且$X \not\rightarrow Y$，A不是X的子集，则称A传递依赖于Y

（1）不满足3NF带来的问题

- R1(<u>S#</u>,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(<u>S#</u>,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#

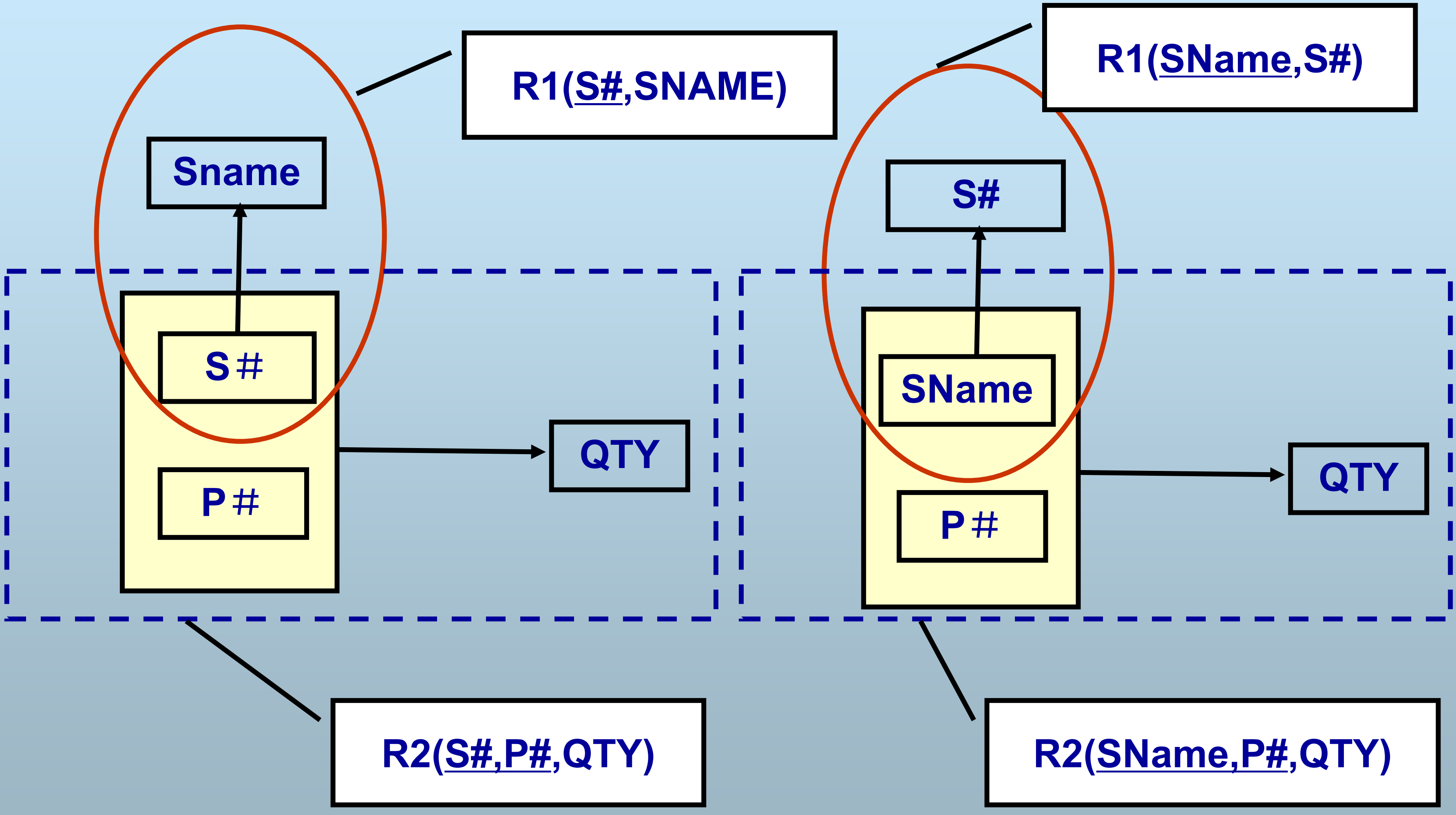
S#	SNAME	P#	QTY
s1	Intel	p1	300
s1	Intel	p2	200
s1	Intel	P3	400
s2	Acer	p1	200

（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）

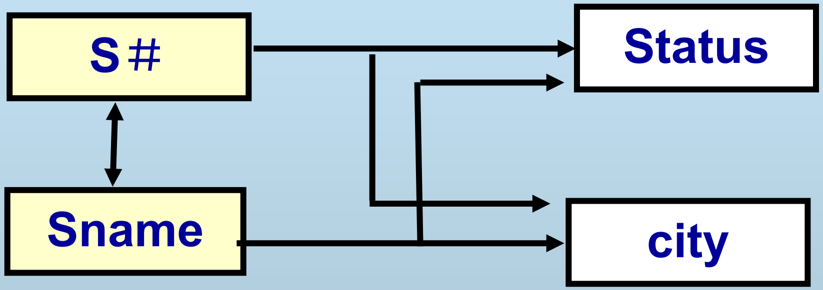


（4）BCNF定义

- 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

（5）BCNF例子1

- R(S#,SNAME,STATUS,CITY)
- 设Sname唯一



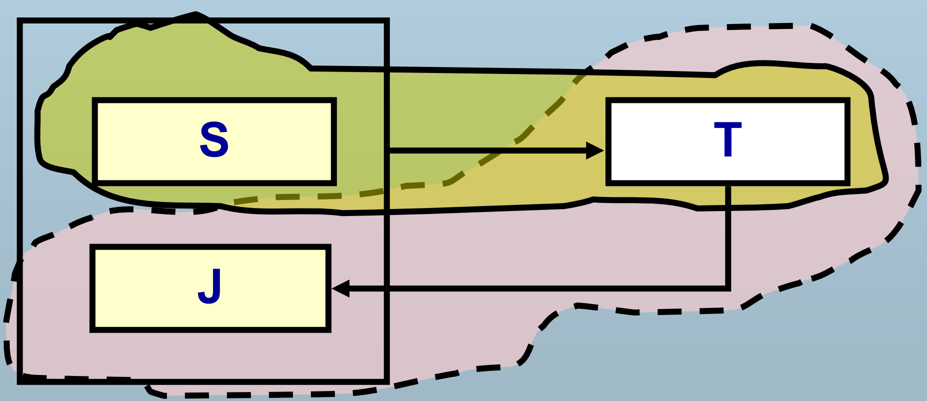
Sname →city,
S# →city,
S# →Sname,
Sname →S#,
Sname→Status,
S#→Status

- 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)}
  - R1中{S#,C#} →G不满足BCNF，分解R1为R3和R4
    ρ:={R3(S#,C#,G), R4(S#,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