

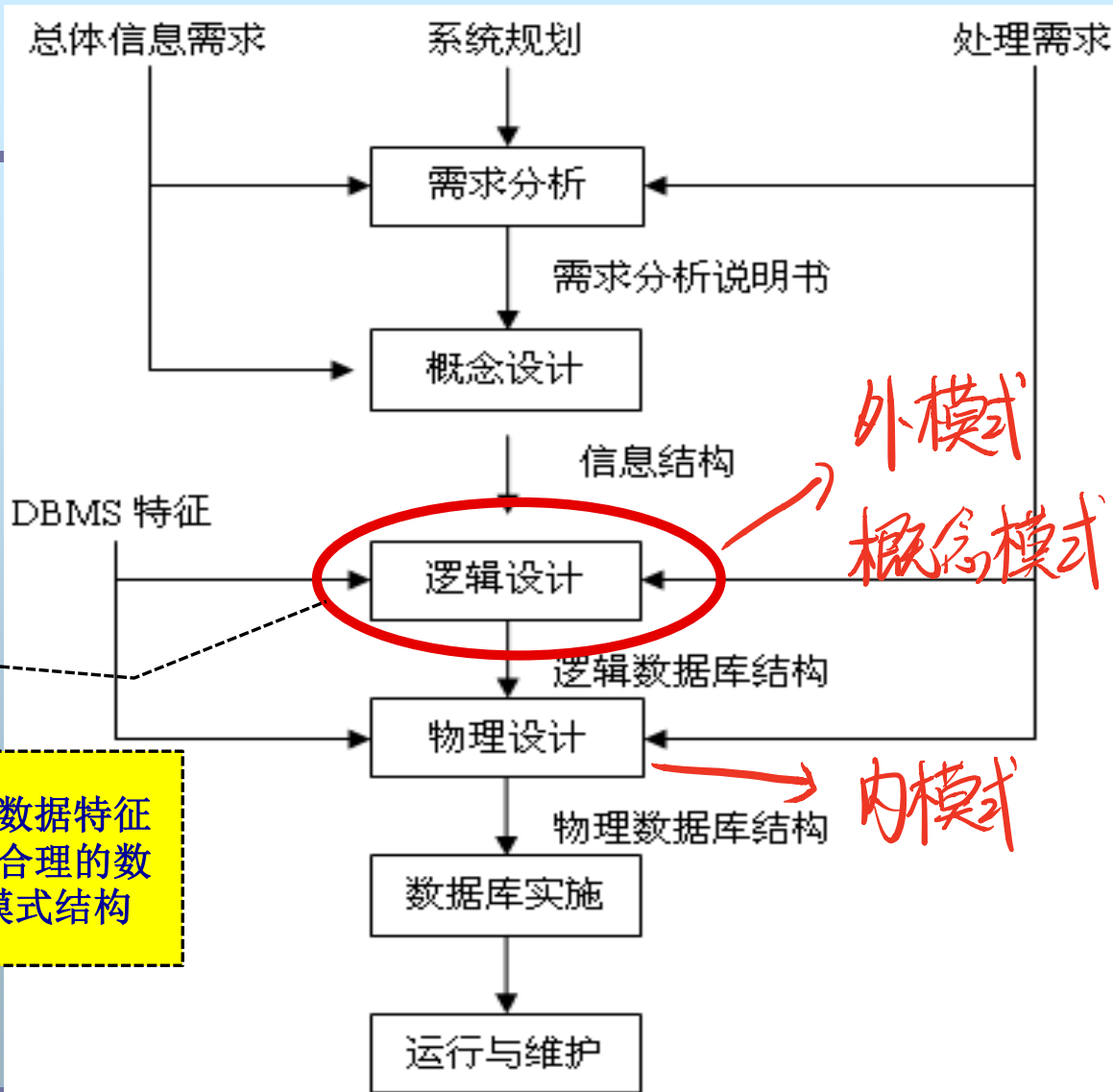
第3章 数据库设计

Part 1: 模式设计

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
- 数据库的模式设计理论是数据库逻辑设计的理论基础

① 功能
② 性能



本章主要内容

- 关系模式的设计问题
- 关系模式的分解
- 关系模式的范式
- 模式分解的算法

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（户口所在地）

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

如何解决？

■ 方法：模式分解

● 方法1：R分解为

◆ $R1(\underline{Tname}, Addr)$

◆ $R2(\underline{C\#}, Cname)$

授课信息丢失了

● 方法2

◆ $R1(Tname, Addr, \underline{C\#})$

◆ $R2(\underline{C\#}, Cname)$

R1中问题依然存在

● 方法3

◆ $R1(\underline{Tname}, Addr)$

◆ $R2(Tname, \underline{C\#}, Cname)$

基本解决问题，但又带来连接查询代价

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、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

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



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**

4、无损连接概念

- 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

4、无损连接概念

- 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 $r = m_{\rho}(r)$

4、无损连接概念

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

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

5、无损连接的测试

- 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： $R(A, B, C), F = \{A \rightarrow B\}$

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

- $\rho2$ 是无损连接， $\rho1$ 不是

5、无损连接的测试

■ 如果R分解成 $n(n > 2)$ 个关系模式

● Chase方法

- ◆ 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
 - 具体参考
《数据库系统概念》或《数据库系统概论》

6、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：

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

函数依赖集新
则是闭包新

(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?
 - $R1 \cap R2 = \{\text{zip}\}, R2 - R1 = \{\text{city}\}, \text{zip} \rightarrow \text{city}$
 - 是无损连接
- 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

- $R(\text{city, street, zip})$,
 $F = \{(\text{city, street}) \rightarrow \text{zip}, \text{zip} \rightarrow \text{city}\}$
- 分解为 $p = \{R1(\text{street, zip}), R2(\text{city, zip})\}$
- 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 $(\text{city, street}) \rightarrow \text{zip}$, 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 一定要保持

- ◆ 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

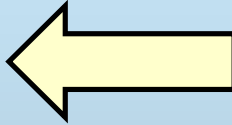
● 保持函数依赖 不一定保持，可通过触发器(写程序)

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

● 既无损连接，又保持函数依赖

Where are we?

- 关系模式的设计问题
- 关系模式的分解
- 关系模式的范式
- 模式分解的算法



三、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?

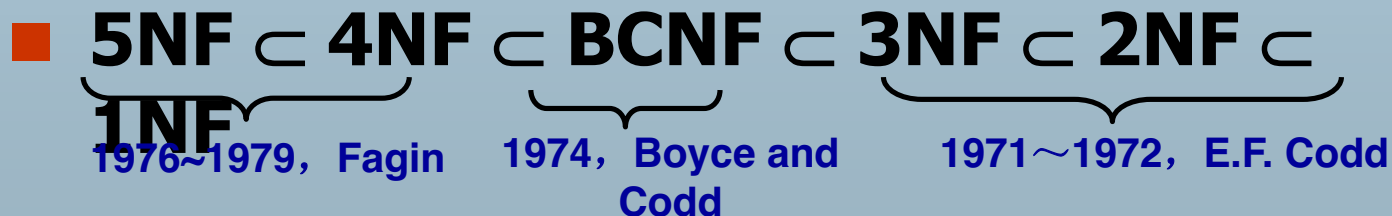
先讲2066模式, 再进行优化

1、范式的概念

■ 范式：满足特定要求的模式

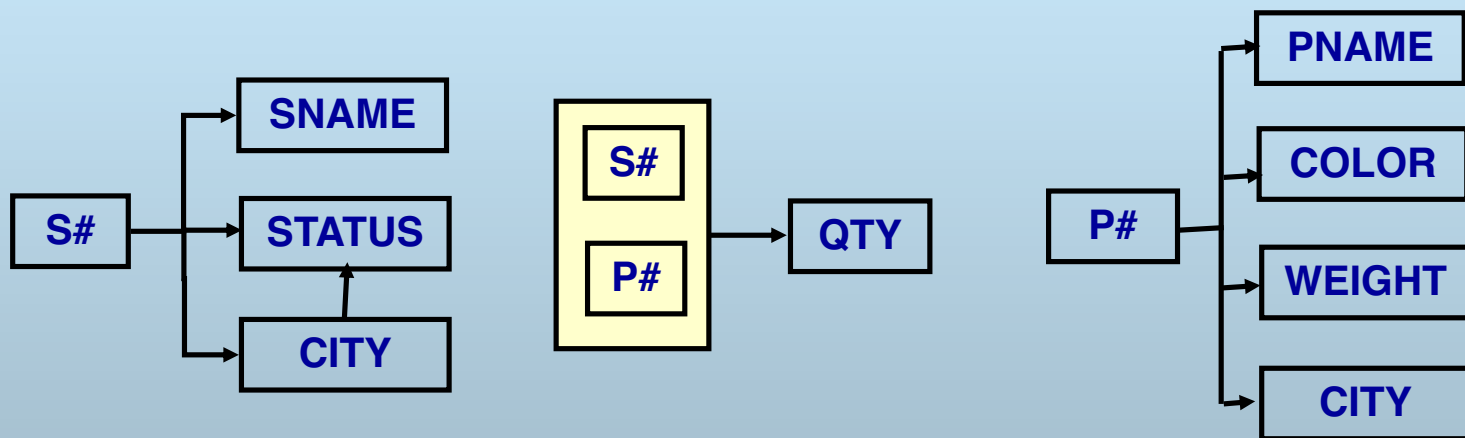
- 不同级别的范式要求各不相同
- 范式可以作为衡量一个关系模式好坏的标准
- 若关系模式R满足范式xNF，记 $R \in xNF$

■ 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的不可约FD集，F可用函数依赖图表达



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

4、2NF

- 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

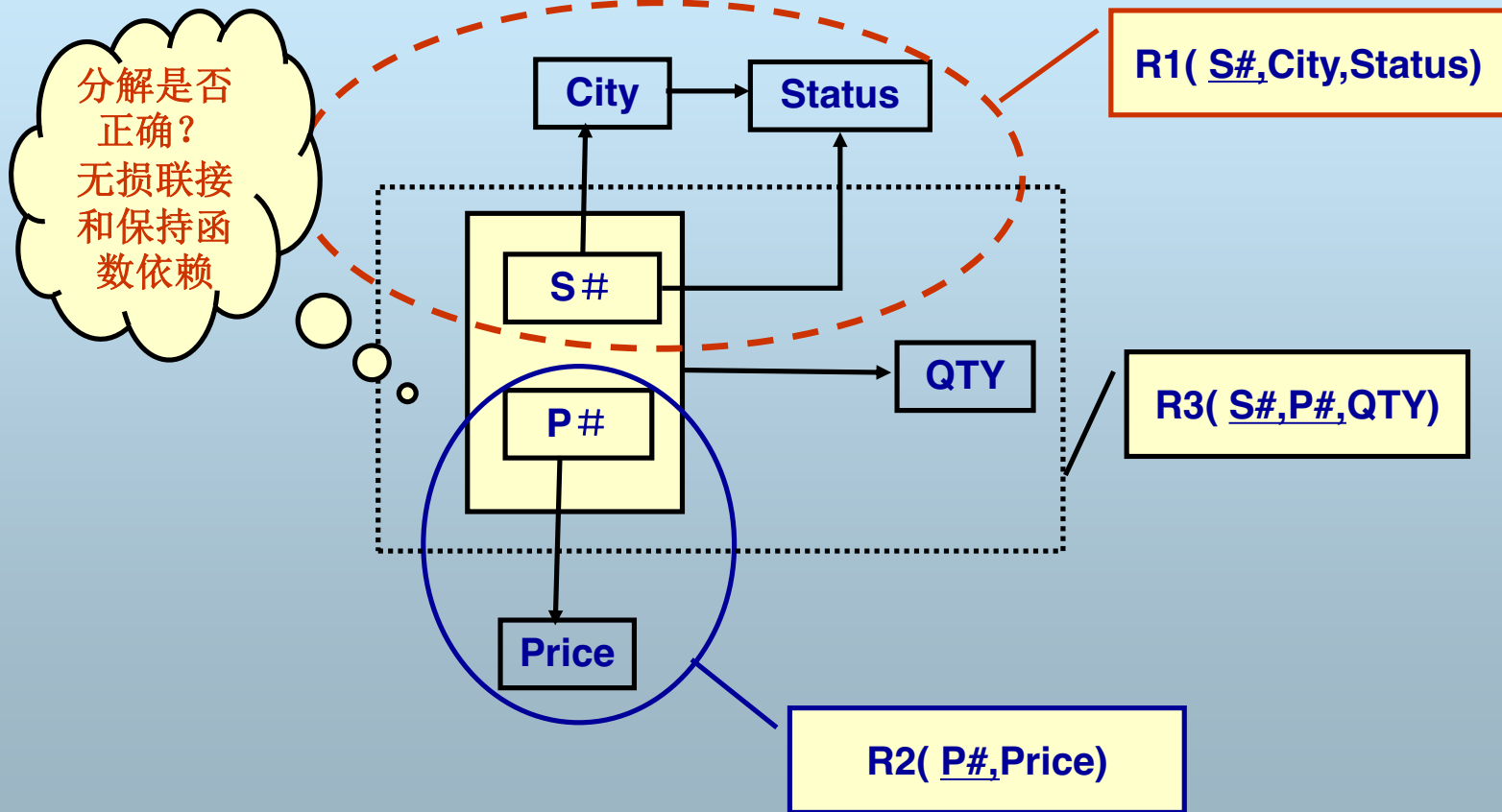
- $R(S\#, P\#, city, status, Price, QTY)$
- $F = \{S\# \rightarrow city, S\# \rightarrow status, P\# \rightarrow Price, city \rightarrow status, \{S\#, P\# \} \rightarrow QTY \}$
- 所以主码为 $\{S\#, P\# \}$
- 但 $city$ 和 $Price$ 都局部函数依赖于主码
- 所以 $R \notin 2NF$

(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

- 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

R1(S#, City, Status)

$F = \{S\# \rightarrow \text{city}, \text{city} \rightarrow \text{status}\}$

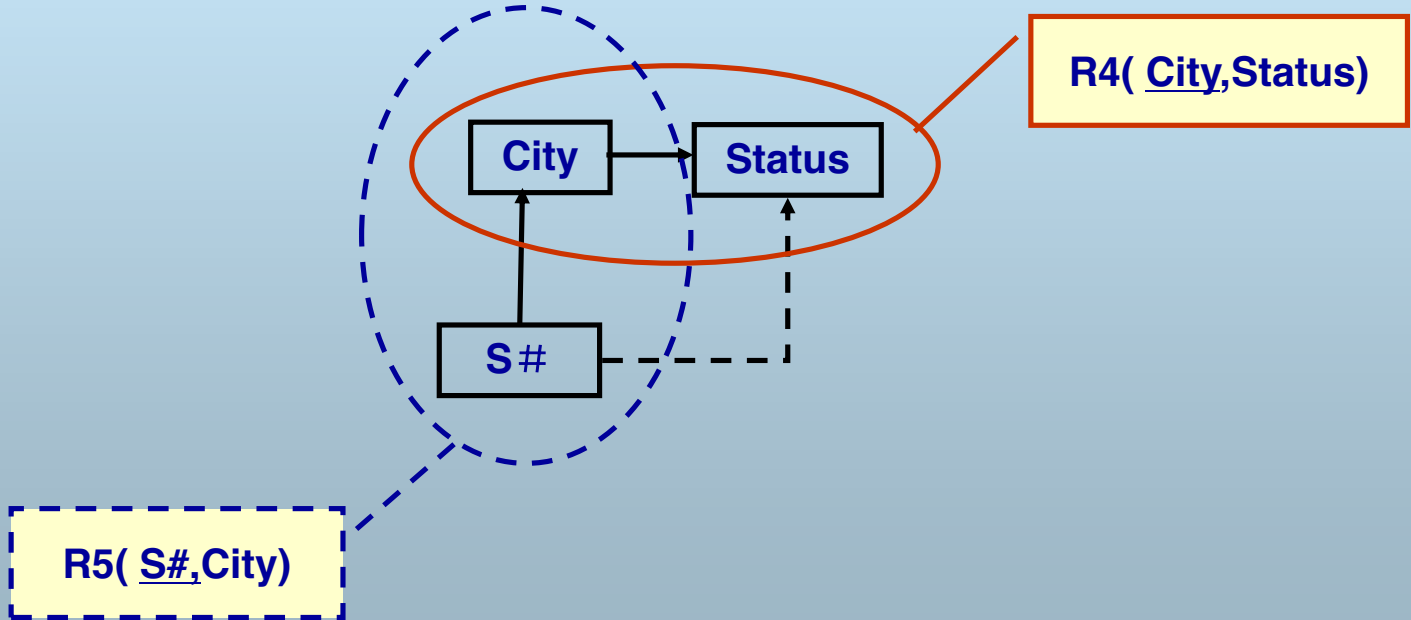
(1) 不满足3NF带来的问题

■ 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，但一般情况下R可能有多个候选码，并且不同的候选码之间还可能相互重叠
- 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

主属性

非主属性

- {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

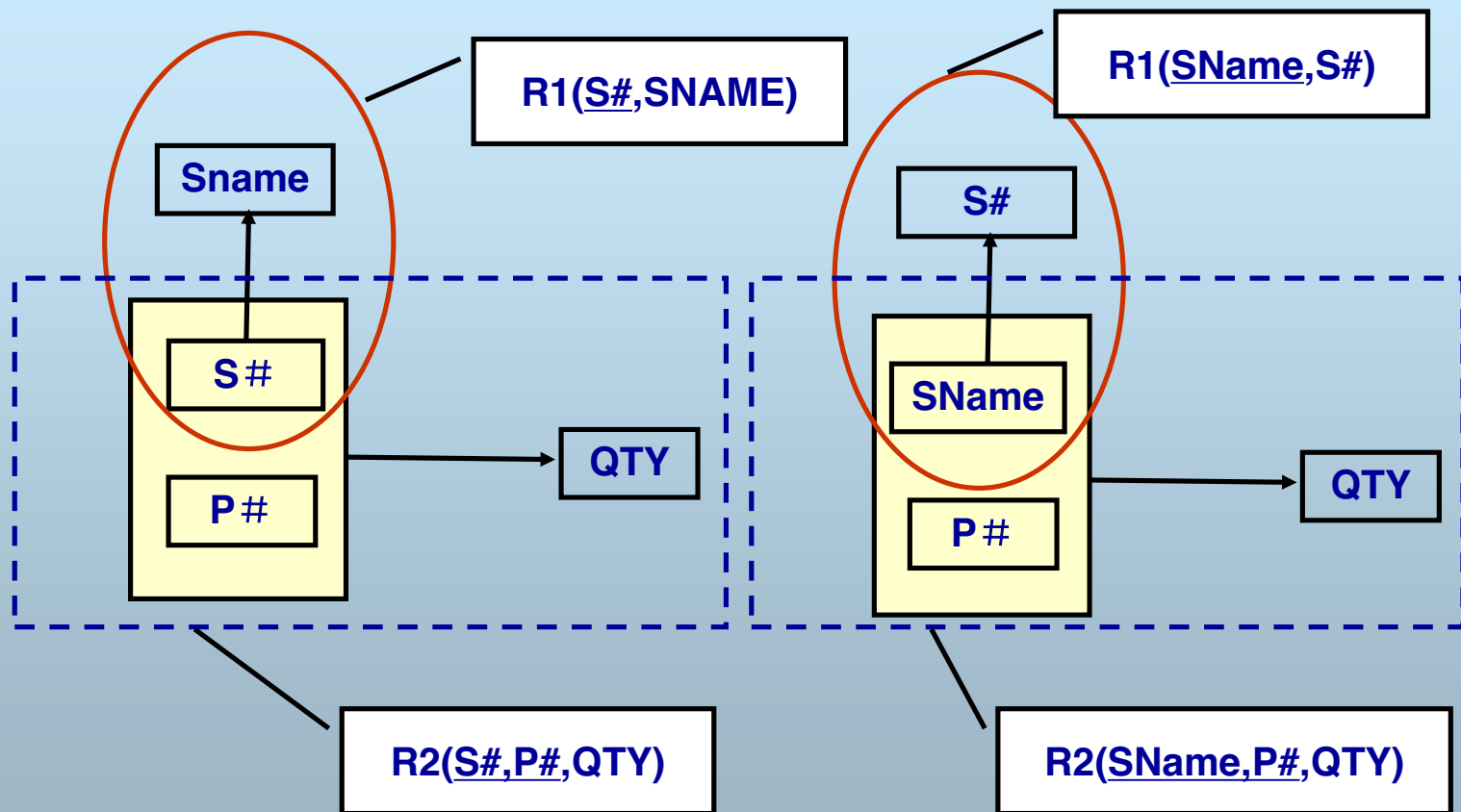
判别是否为 3NF
没有用到。
因为右边为主属性

(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

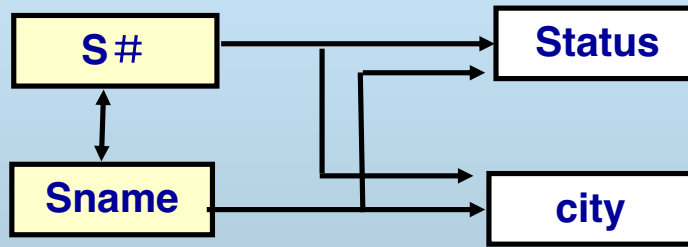
- (C.J. Date) 如果关系模式R的所有不平凡的、完全的函数依赖的决定因素（左边的属性集）都是候选码，则 $R \in \text{BCNF}$
- (萨 & 王) 关系模式 $R \in 1\text{NF}$ ，若R中的任一函数依赖 $X \rightarrow Y$ 且 $Y \not\subseteq X$ 时X必包含候选码，则 $R \in \text{BCNF}$
 - 3NF: 不允许非主属性到非码的FD，但允许主属性到其它属性的FD
 - BCNF: 不允许主属性、非主属性到非码的FD

C.J.Date, An Introduction to Database Systems

萨 & 王, 数据库系统概论

(5) BCNF例子1

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



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

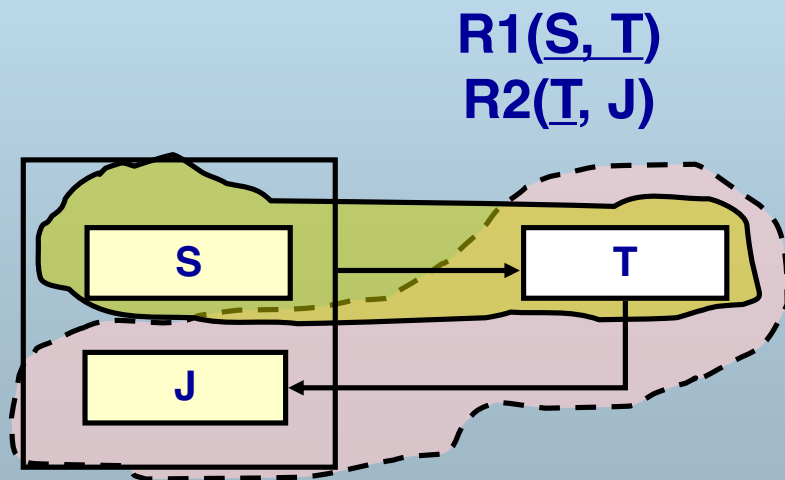
- BCNF模式的函数依赖图中，箭头都是从候选码中引出，所有不平凡FD的左边都是候选码

3NF无法约束的是左端有主属性

(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 ^码
- R 不属于 BCNF

左端必须是候选码



分解到 BCNF 不一定能保持函数依赖

四、模式分解的几个算法

- 保持函数依赖地分解到3NF的算法
- 无损并且保持函数依赖分解为3NF的算法
- 无损分解为BCNF的算法

1、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 ）(手2求)
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

(1)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: $R1(AB), R2(ACE)$
- $p = \{R'(DF), R1(AB), R2(ACE)\}$

2.无损连接且保持函数依赖地分解到3NF

1. 首先用算法1求出R的保持函数依赖的3NF分解, 设为 $q=\{R_1, R_2, \dots, R_k\}$
2. 设X是R的码, 求出 $p=q \cup \{R(X)\}$
3. 若X是q中某个 R_i 的子集, 则在p中去掉 $R(X)$
4. 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

3、无损连接地分解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 $S2(U_s - A)$, 此分解是无损连接的
 $// (\{XA\} \cap \{U_s - A\} = X) \rightarrow (A = \{XA\} - \{U_s - A\})$
 - 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

(1)例子

分解为BCNF不一定满足函数依赖

■ $R(S\#,C\#,G,TN,D),$

$F=\{\{S\#,C\#\} \rightarrow G, C\# \rightarrow TN, TN \rightarrow D\}$

1. $p=\{R\};$ (C# → TN 先开始 则 $R_1(C\#,TN)$)
2. $TN \rightarrow D$ 不满足BCNF定义, 分解R $R_2(S\#,C\#,G,D)$
 $p=\{R_1(S\#,C\#,G,TN), R_2(TN,D)\}$ $R_3(S\#,C\#,G)$
 $R_4(S\#,C\#,D)$
3. R_1 中 $C\# \rightarrow TN$ 不满足BCNF, 分解 R_1 为 R_3 和 R_4
 $p=\{R_3(S\#,C\#,G), R_4(C\#,TN), R_2(TN,D)\}$
4. p 中各模式均满足BCNF, 结束

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
- 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