第4章 数据依赖与关系模式规范化

4.1 问题的提出

4.2 数据依赖

4.3 关系模式分解

4.4 关系模式规范化

4.1 关系模式设计中的问题

- 前面我们已经讨论了关系数据库的基本概念、关系模型的三个部分以及关系数据库的标准语言SQL
- 但是，还有一个基本问题尚未涉及，即针对一个具体的应用问题，应该如何构造一个适合于它的关系数据库模式？应该构造几个关系模式？每个关系由哪些属性组成？
- 本章的任务是研究设计一个“好”的（即没有“毛病”的）关系模式的方法

实例分析

- 前面章节中多次引用过的三个基本表：

  Student(S#, Class, Sname, …)、

  Course(C#, Cname, Teacher)、

  SC(S#, C#, Grade)

- 对于这个应用而言，为什么仅设计了这三个表，如果将这三个表合并成一个或两个表，或者将其拆分为更多的表，这样是否更合适？

实例分析

- 考虑下面的关系模式SCT:

SCT(S#, C#, GRADE, Teacher, Dept)

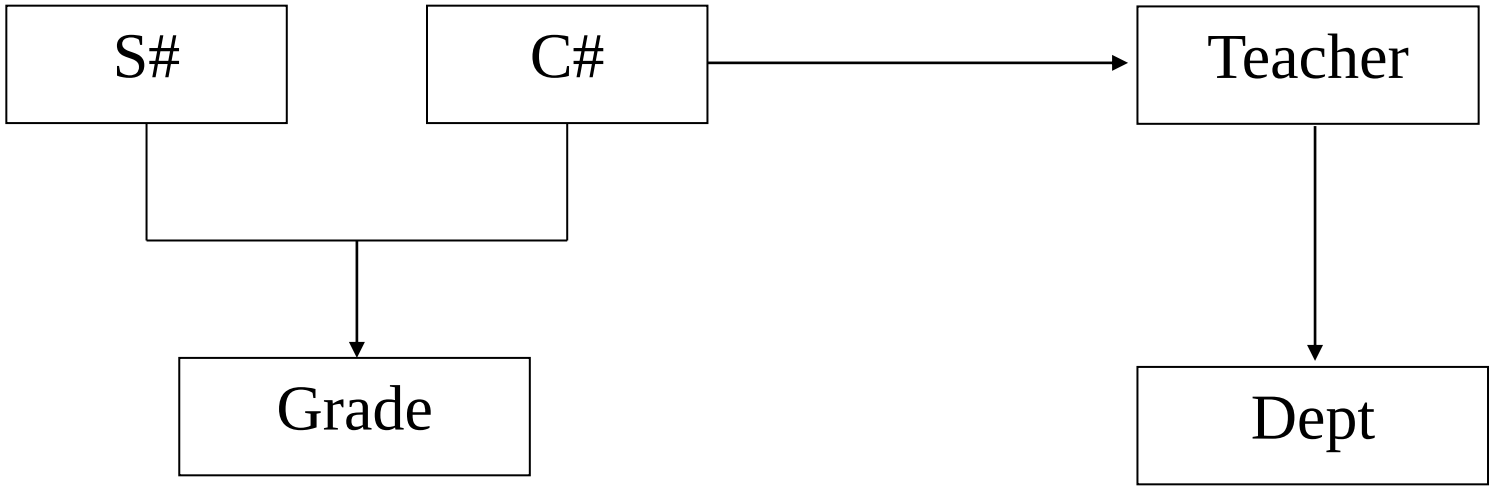


图4.1关系模式SCT属性间依赖关系

实例分析

S#	C#	Grade	Teacher	Dept
02055019	CS－01	89	刘朝	计算机
02055019	CS－03	86	张莹	计算机
02055037	CS－01	68	刘朝	计算机
02055066	CS－02	92	林琳	计算机
02055066	CS－03	73	张莹	计算机
…	…	…	…	…

图4.2 关系模式SCT实例

实例分析

- **关系模式SCT存在的问题：**

  1. 数据冗余(Redundancy)
  2. 更新异常(Update Anomalies)
  3. 插入异常(Insertion Anomalies)
  4. 删除异常(Deletion Anomalies)

实例分析

- **结论：**
  - SCT不是一个”好”的关系模式
  - “好”的模式：数据冗余应尽可能少，不会发生插入异常、删除异常、更新异常
- **原因：**
  - 由存在于模式中的某些”数据依赖”引起的
- **解决方法：**
  - 通过分解关系模式来消除其中不合适的数据依赖

4.2 数据依赖

- 数据依赖(Data Dependency)是通过一个关系中各**属性值相等与否**体现出来的一种数据间的相互制约关系，是现实世界中事物属性间相互联系的抽象，是数据内在的性质，是**语义**的体现
- 人们已经提出了许多种类型的数据依赖，其中最重要的是**函数依赖**(Functional Dependency简记为FD)和**多值依赖**(Multivalued Dependency简记为MVD)

常用符号及其含义

- R表示一个抽象的关系模式，R上全部函数依赖的集合记为F；
- R的属性全集一般用U表示，U＝{A1，A2，…，An}，其中的每一个 Ai（1≤i≤n）均表示R的一个属性，X、Y、Z等表示U的一个真子集，也就是若干属性的集合；

关系的内涵和外延

- r是关系模式R上的一个实例，记为R(r)，即满足R的一些元组的集合，R可以有多个不同的实例r
- 通常关系模式很少变化，经常发生变化的是关系的实例，因此将关系的模式称为关系的内涵，而将关系的实例r称为关系的外延，由于用户经常对关系进行插入、删除和修改操作，因此外延是与时间有关的，随着时间的推移在不断变化

关系的内涵和外延

- 关系的内涵是对数据的定义以及数据完整性约束的定义，一般是与时间独立的
- 对数据的定义包括对关系、属性、域的定义和说明。对数据完整性约束的定义涉及面较广，主要包括以下几个方面：
  - 静态约束，涉及到数据之间联系（称为“数据依赖，data dependences）、主键和值域的设计
  - 动态约束，定义各种操作（插入、删除、修改）对关系值的影响

函数依赖的定义

- 定义4.1：设R为关系模式，r是R上的任意一个关系实例，X，Y⊆U是R的两个属性子集，若对于r上的任意两个元组t1，t2∈r都有：如果 t1[X]＝t2[X]，则必有t1[Y]=t2[Y]，则称在R上X函数决定Y或者Y函数依赖于X，记为X→Y，X称为决定子(Determinant)

- 例：

A	*B*	*C*	*D*
*a*1	*b*1	*c*1	*d*1
*a*1	*b*1	*c*2	*d*2
*a*2	*b*2	*c*3	*d*3
*a*3	*b*1	*c*4	*d*4

A	*B*	*C*	*D*
*a*1	*b*1	*c*1	*d*1
*a*1	*b*2	*c*2	*d*2
*a*2	*b*2	*c*3	*d*3
*a*3	*b*2	*c*4	*d*4

- 分别检验：A→C？ C→A？ AB→D？
- 辨识：
  - 满足依赖的关系：依赖在模式的某个关系实例上成立
  - 模式上成立的依赖：依赖在模式的所有关系实例上都成立

函数依赖的定义

- 函数依赖是语义范畴的概念，它反映了一种语义完整性约束，只能根据语义来确定一个函数依赖是否成立
- 例如，“姓名”→“年龄”这个函数依赖只有在没有同名人的条件下成立，否则，此函数依赖不成立
- 函数依赖是指关系R模式的所有关系元组均应满足的约束条件，而不是关系模式中的某个或某些元组满足的约束条件

函数依赖的定义

- 定义4.2：设R为关系模式，X、Y是R的不同属性集，如果X→Y成立，且不存在X’⊂ X使得X’ →Y也成立，则称Y**完全函数依赖**于X，记为$X \xrightarrow{f} Y$，否则称Y**部分函数依赖**于X，记为$X \xrightarrow{P} Y$
- 完全函数依赖中的决定子不包含冗余属性，即只要将决定子中的任何一个属性去掉，这个函数依赖就不再成立了
- 定义4.3：如果$Y \subseteq X$, 则称X→Y为**平凡函数依赖**

函数依赖的定义

- 定义4.4：设X、Y、Z是R上的不同属性集合，如果有X→Y，Y→Z成立且Y $\not\rightarrow$ X，则称Z**传递函数依赖**于X
- 条件Y $\not\rightarrow$ X非常重要，如果没有这个条件，那么X与Y就是一一对应关系，从而X→Z就是直接函数依赖

函数依赖的逻辑蕴涵

- 定义4.5：设F是在关系模式R上成立的函数依赖的集合，X→Y是一个函数依赖，如果对于R的每个满足F的关系r也满足X→Y，那么称F**逻辑蕴涵**X→Y，记为F ⊨ X→Y
- 例如：{X→Y，Y→Z} ⊨ X→Z

函数依赖集合的闭包

- 定义4.6：由函数依赖集合F所逻辑蕴涵的全部函数依赖所构成的集合称之为**F的闭包**，记作F^+，即

  $F^+ = \{ X \rightarrow Y \mid F \models X \rightarrow Y \}$

- 据定义4.6易知，函数依赖集的闭包F^+具有以下特点：

  (1) $F \subseteq F^+$，这是因为根据闭包的定义F中的每个函数依赖必定也在中F^+；

  (2) $(F^+)^+ = F^+$，该性质说明闭包运算是幂等的，即F经过任意多次的闭包运算后其结果仍然等于F^+

  (3) 如果$F = F^+$，则称F是完备的

函数依赖集合的闭包

- 例4.1.计算R(A，B，C)上的函数依赖集合F＝{ A→B，B→C}的闭包：

$$F^{+}=\left\{\begin{array}{l} A\rightarrow B，\ B\rightarrow C，\ A\rightarrow C \\ A\rightarrow A，\ B\rightarrow B，\ C\rightarrow C \\ AB\rightarrow A，\ AB\rightarrow B，\ AB\rightarrow AB... \end{array}\right\}$$

- F^{+}的计算是一个NP完全问题

函数依赖的推理规则

- 设U是关系模式R的属性集，F是R上的函数依赖集，则有：
  - A1（自反率）：如果$Y \subseteq X \subseteq U$，则X→Y成立；
  - A2（增广率）：如果X→Y成立，且$Z \subseteq U$，则XZ→YZ成立
  - A3（传递率）：如果X→Y，Y→Z成立，则X→Z成立
- 人们又把自反律(A1)，传递律(A2)和增广律(A3)称为Armstrong公理

函数依赖的推理规则

- 引理4.1：FD推理规则A1、A2和A3是正确的
- 也就是，如果X→Y是从F用推理规则导出，那么X→Y在F^+中
- 由A1～A3易推出下面的三条推理规则也是正确的：

(1) 合并规则：若X→Y，X→Z成立，则X→YZ成立；

(2) 伪传递规则：若X→Y，WY→Z成立，则WX→Z成立；

(3) 分解规则：若X→Y，且Z⊆Y，则有X→Z；

FD的逻辑导出

- 定义4.7：给定关系模式R<U,F>，如果能由F根据Armstrong公理导出X→Y，则称X→Y是F的**逻辑导出**，记为F=> X→Y

属性集合的闭包

- 定义4.8：**属性集X**关于R(U，F)上的函数依赖集合**F的闭包** X_F^+ 定义为 X^+_F ＝{A|A∈U， F=> X→A}
- 根据A1（自反率）可知对于任一属性集X一定有 $X \subseteq X^+_F$ 此外，计算 X^+_F 的工作量应该远远小于计算F^+的工作量，因为 X^+_F 中的属性一定是可数的，最多就是属性全集U而已

属性集合的闭包

引理 4.2：$X \rightarrow Y$ 可由 Armstrong 公理推出的充分必要条件是 $Y \subseteq X^{+}_{F}$。

证：充分性：已知 $Y \subseteq X^{+}_{F}$ 成立，要证 $X \rightarrow Y$ 可由 Armstrong 公理推出。设 $Y = B_1B_2...B_n$，则有 $B_1B_2...B_n \subseteq X^{+}_{F}$，于是有 $B_i \in X^{+}_{F}$ $(1 \leqslant i \leqslant n)$，按照定义 4-6 可知 $X \rightarrow B_i$ 可由 Armstrong 公理推出，再根据合并规则及 i 的任意性可得 $X \rightarrow B_1B_2...B_n$ 可由 Armstrong 公理推出，即 $X \rightarrow Y$ 可由 Armstrong 公理推出；

必要性：已知 $X \rightarrow Y$ 可由 Armstrong 公理推出，要证 $Y \subseteq X^{+}_{F}$。类似于充分性的证明，利用分解规则可证。

属性集的闭包

- 算法4.1：求属性集X(X⊆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：

  1) 令X(0)=X，i=0

  2) 求B，这里B={A|(存在V→W)(V→W∈F∧V∈X(i) ∧ A∈W)}；

  3) X(i+1)=X(i)∪B

  4) 判断X(i+1)=x(i)吗?

  5) 若相等或X(i)=U 则X(i)就是X^{+}_{F},算法终止。

  6) 若否，则i=i+1，返回第 2) 步。

属性集的闭包

- 例4.2: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 $(AB)^{+}_{F}$ 。

- 解：由算法，X(0)=AB；计算X(1)； 逐一的扫描F集合中各个函数依赖,找左部为A,B,或AB的函数依赖。得到两个：AB→C，B→D。于是X(1)=AB∪CD=ABCD。

  因为 X(0)≠X(1)，所以再找出左部为ABCD子集的那些函数依赖，又得到C→E，AC→B，于是
  X(2)=X(1)∪BE=ABCDE。
  因为X(2) 已等于全部属性集合,所以$(AB)^{+}_{F}$ =ABCDE

- 对于算法1.1， 令ai=|X(i)|,{ai }形成一个步长大于1的严格递增的序列,序列的上界是|U|,因此该算法最多|U|-|X|次循环就会终止

属性集的闭包

- 例4.3：属性集U为ABCD，
  FD集为｛ A→B，B→C，D→B ｝
- 则用上述算法，可求出：
- A^+=ABC，
  AD^+=ABCD
  BD^+=BCD
  等等

Armstrong公理的正确性和完备性

- **Armstrong公理的正确性**是指“使用推理规则从FD集F推出的FD必定在F^+中”

  即，如果$F=>X\rightarrow Y$ 则 $F|=X\rightarrow Y$

- **Armstrong公理的完备性**是指“F^+中的FD都能从F集使用推理规则集导出”

  即，如果$F|= X\rightarrow Y$ 则 $F => X\rightarrow Y$

- **Armstrong公理的正确性和完备性**保证了FD推导过程的有效性和可靠性

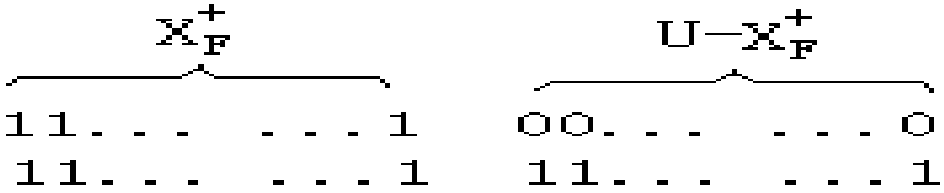
Armstrong公理的正确性和完备性

- **定理4.1：Armstrong公理是完备的**
- 证明：我们证明完备性的逆否命题，即若函数依赖X→Y不能由F从Armstrong公理导出，那么它必然不为F所蕴含，其证明分三步：

1)若V→W成立，且V⊆ X^{+}_{F} ，则W⊆ X^{+}_{F}
因为V⊆ X^{+}_{F} ,所以X→V成立；由A3规则有X→W成立，所以W⊆ X^{+}_{F}

Armstrong公理的正确性和完备性

2) 由下列两个元组构成的二维表，必是R<U,F>的一个关系r，即满足F中的全部函数依赖。



若r不是R<U,F>的关系，则必由于F中有函数依赖V→W在r上不成立所致,由r的构成可知，V必定是X^+_F的子集，而W不是X^+_F的子集，但是由 1)$W \subseteq X^+_F$ ，矛盾!所以r必是R<U,F>的一个关系

Armstrong公理的正确性和完备性

3) 若X→Y不能由F从Armstrong公理导出，则Y不是X^{+}_{F}的子集，因此必有Y的子集Y‘满足Y’ $\subseteq$ U- X^{+}_{F} ，即X→Y在r上不成立，故X→Y必不为F蕴含。

证毕

- **Armstrong公理的正确性和完备性**说明“逻辑导出”与“逻辑蕴含”是两个等阶的概念
- 因此,F^{+}也可以说成是由F出发借助Armstrong公理导出的函数依赖的集合
- 从蕴含(或导出)的概念出发，又引出了两个函数依赖集等价和最小依赖集的概念

函数依赖集的等价覆盖

- **定义4.8：设F，G是两个函数依赖集合，如果 $F^+=G^+$ ，则称F等价于G，或者F与G互相覆盖**
- **引理4.3： F与G等价的充分必要条件是$F\subseteq G^+$ 并且$G\subseteq F^+$**
- **证：必要性显然，只需证充分性：**
  1） 若$F \subseteq G^+$ ，则$X^+_F \subseteq X^+_{G+}$
  2） 任取$X\rightarrow Y\in F^+$ 则有 $Y \subseteq X^+_F \subseteq X^+_{G+}$ 。 所以 $X\rightarrow Y\in (G^+)^+ = G^+$ ，即$F^+\subseteq G^+$
  3） 同理可证$G^+ \subseteq F^+$ ，所以$F^+=G^+$

函数依赖集的等价覆盖

- 要判定$F \subseteq G^+$，只须逐一对F中的函数依赖X→Y，考查Y是否属于X^+_{G+}
- 引理4.3给出了判断两个函数依赖集等价的可行算法

最小函数依赖集

- 定义4.9：若F满足下列条件，

(1)F中所有函数依赖的右部均为单属性；

(2)F中不存在这样的函数依赖X→A及Z⊂ X，使得

$$F^{+} = ((F - \{X \rightarrow A\}) \cup \{Z \rightarrow A\})^{+}$$ ；

(3)F中不存在这样的函数依赖X→A：使得

$$F^{+} = (F - \{X \rightarrow A\})^{+}$$

则称F为**最小函数依赖集**或最小覆盖

最小函数依赖集

- 定理4.2：每一个函数依赖集F均等价于一个最小函数依赖集Fmin
- 证：这是一个构造性的证明，我们分三步对F进行“极小化处理”，找出F的一个最小依赖集来。
  1. 逐一检查F中各函数依赖FDi：X→Y，若Y=A1A2 ...Ak ，k>2，则用{X→Aj |j=1,2,…k}来取代X→Y
  2. 逐一取出F中各函数依赖FDi:X→A，设X=B1B2 ...Bm ，逐一考查Bi （i=1,2,...,m)，若A∈(X-Bi)^+_F ，则以X-Bi 取代X
     注：因为F与F-{X→A}∪{Z→A}等价的充要条件是A∈Z^+ 其中Z=X-Bi
  3. 逐一检查F中各函数依赖FDi:X→A，令G=F-{X→A}，若A∈X^+_G，则从F中去掉此函数依赖

     注：因为F与G等价的充要条件是A∈X^+_G
- 最后剩下的F就一定是极小依赖集，并且与原来的F等价，因为，我们对F的每一次‘改造’都保证了改造前后的两个函数依赖集等价
- 应当指出，F的最小依赖集Fmin不一定是唯一的，它和我们对各函数依赖FDi 及X→A中X各属性的处置顺序有关

最小函数依赖集

例4.4: R(A,B,C,D,E,H,I),

F = {A→BE, AH→D, B→C, BD→E, C→D, H→I,I→H,
H→BE}

试求F的最小依赖集Fmin。

解：(1)右部拆成单属性

F={A→B, A→E ,AH→D, B→C, BD→E, C→D, H→I,I→H,
H→B, H→E}

(2)考察左部不是单属性的函数依赖，消除多余属性

AH→D ∵$((AH)-H)_F^+$=ABECD,D∈$((AH)-H)_F^+$

∴以A→D取代AH→D

BD→E ∵$((BD)-D)_F^+$=BCDE,E∈$((BD)-D)_F^+$

∴以B→E取代BD→E

F={A→B, A→E ,A→D, B→C, B→E, C→D, H→I,I→H,
H→B, H→E}

最小函数依赖集

(3)消除多余的函数依赖

A→B ∵A_G^+=AED, B∉ A_G^+(G=F-{A→B}) ∴保留该函数依赖

A→E ∵A_G^+=ABCDE, E∈ A_G^+(G=F-{A→E}) ∴不保留该函数依赖

A→D ∵$(A)_G^+$=ABCDE, D∈ $(A)_G^+$(G=F-{A→D}) ∴不保留该函数依赖

B→C ∵B_G^+=B, C ∉ B_G^+(G=F-{B→C}) ∴保留该函数依赖

B→E ∵$(B)_G^+$=BCD, E ∉ $(B)_G^+$(G=F-{B→E}) ∴保留该函数依赖

C→D ∵C_G^+=AE, D∉C_G^+(G=F-{C→D}) ∴保留该函数依赖

H→I ∵H_G^+=HBECD, I ∉H_G^+(G=F-{H→I}) ∴保留该函数依赖

I→H ∵I_G^+=I, H ∉ I_G^+(G=F-{I→H}) ∴保留该函数依赖

H→B ∵H_G^+=HIE, B∉ H_G^+(G=F-{H→B}) ∴保留该函数依赖

H→E ∵H_G^+=HBCDE, E∈H_G^+(G=F-{H→E}) ∴不保留该函数依赖

最后剩下的F是最小函数依赖集，

F_{min}= {A→B, B→C, B→E, C→D, H→I, I→H, H→B}

最小函数依赖集

- 应当指出，F的最小依赖集Fmin不一定是唯一的，它和我们对各函数依赖FDi 及X→A中X各属性的处置顺序有关

多值依赖

- 除了函数依赖以外，关系的属性间还存在其他一些数据依赖关系，多值依赖(multivalued dependency,MVD)就是其中之一

- 例1： 学校中某一门课程由多个教员讲授，他们使用相同的一套参考书。每个教员可以讲授多门课程，每种参考书可以供多门课程使用。我们可以用一个非规范化的关系来表示教员T,课程C和参考书B之间的关系：

课程C	教员T	参考书B
物理	李 勇	普通物理学
	王 军	光学原理
		物理习题集
数学	李 勇	数学分析
	张 平	微分方程
		高等代数

多值依赖

- **把这张表变成一张规范化的二维表,就成为:Teaching**
- 课程C　　教员T　　参考书B

课程C	教员T	参考书B
物　理	李　勇	普通物理学
物　理	李　勇	光学原理
物　理	李　勇	物理习题集
物　理	王　军	普通物理学
物　理	王　军	光学原理
物　理	王　军	物理习题集
数　学	李　勇	数学分析
数　学	李　勇	微分方程
数　学	李　勇	高等代数
数　学	张　平	数学分析
数　学	张　平	微分方程
数　学	张　平	高等代数

多值依赖

- 仔细考察这类关系模式，发现它具有一种称为多值依赖(MVD)的数据依赖
- 定义4.10：设R(U)是属性集U上的一个关系模式。X，Y，Z是的U的子集,并且Z=U-X-Y。关系模式R(U)中多值依赖X→→Y成立，当且仅当对R(U)的任一关系r,给定的一对(x，z)值有一组Y的值，这组值仅仅决定于x值而与z值无关
- 例如，在关系模式TEACHING中，对于(物理，光学原理)有一组T值{李勇，王军}，这组值仅仅决定于课程C上的值(物理)，也就是说对于另一个(物理,普通物理学)它对应的一组T值仍是{李勇,王军}，尽管这时参考书B的值已经改变了。因此，T多值依赖于C，即C→→T

多值依赖

- 以上对多值依赖进行了一些直观的讨论，下面对MVD进行形式化描述
- 定义4.11：设R(U)是属性集U上的一个关系模式。X，Y，Z是的U的子集,并且Z=U-X-Y，如果对R(U)的任一关系r，都有如下性质：

  如果r中存在2个元组s、t，使得：

  s[X]=t[X]

  则r中必存在元组u，使得：

  (1) s[X]=t[X]=u[X]

  (2) u[Y]=t[Y] 且 u[Z]=s[Z]

  (即交换s、t在Y上的值得到的2个元组必在r中)

  则称关系模式R满足多值依赖X→→Y

多值依赖

- 一般,当关系至少有三个属性,其中的两个是多值时,且它们的值只依赖于第三个属性时,才会有多值依赖;
- 如果关系中存在非平凡非FD的多值依赖则会导致数据冗余和更新异常等问题,所以要消除异常,必须消除多值依赖,方法是，通过建立两个关系，让每个关系只存储一个多值属性的数据。

MVD举例

对于W的每一个值，S有一个完整的集与它对应不论C取什么值，所以W $\rightarrow\rightarrow$ S

- 关系模式WSC(W,S,C)中，W表示仓库，S表示保管员，C表示商品。假设每个仓库有若干个保管员，有若干种商品。每个保管员保管所在仓库的所有商品，每种商品被所有保管员保管

W	S	C
W1	S1	C1
W1	S1	C2
W1	S1	C3
W1	S2	C1
W1	S2	C2
W1	S2	C3
W2	S3	C4
W2	S3	C5
W2	S4	C4
W2	S4	C5

MVD举例

- 例:设关系模式R(A,B,C)上有一个MVD A $\rightarrow\!\!\rightarrow$ B,如果R中已有（a,b1,c1),(a,b2,c2)，（a,b3,c3),那么这些关系中至少还应该存在哪些元组?

至少还存在6个元组:

(a,b1,c2)(a,b2,c1)(a,b1,c3)

(a,b3,c1)(a,b2,c3)(a,b3,c2)

有关多值依赖的推导规则

- 与函数依赖一样，多值依赖也有一组推导规则：
  - A4：互补律(complementation)
    如果X→→Y，则X→→(U-X-Y)
  - 以后如果需要，可用X→→Y| (U-XY)表示多值依赖，以强调其互补关系
  - A5：扩展律(MVD)
    如果X→→Y且V⊆W，则WX→→VY
  - A6：传递律(MVD)
    如果X→→Y且Y→→Z，则X→→(Z-Y)
  - 下面两条为(FD+MVD)公理：
  - A7：如果X→Y，则X→→Y，即FD是MVD的特例。
  - A8：如果X→→Y、Z⊆Y且对某一个与Y不相交的W有：如果 W→Z，则X→Z。

有关多值依赖的推导规则

- 由上述公理，还可以得出下列四个有关MVD的推导规则：

1)MVD合并规则

如果X→→Y、X→→Z，则X→→YZ

2)MVD伪传递规则

如果X→→Y、WY→→Z，则WX→→(Z-WY)

3)混合伪传递规则

如果X→→Y、XY→Z，则X→(Z-Y)

4)MVD分解规则

如果X→→Y、X→→Z，则X→→(Y∩Z)、X→→(Y-Z)
X→→(Z-Y)均成立。

- 以上规则的证明从略

多值依赖的性质

- 定义4.12：若X→→Y，而Z=U-XY为空，则称X→→Y 为平凡的多值依赖
- 多值依赖具有以下性质：
  1）多值依赖具有对称性。即若X→→Y，则X→→Z，其中Z＝U－XY。
  2）多值依赖的传递性。即若X→→Y，Y→→Z， 则X→→Z－Y。
  3）函数依赖可以看作是多值依赖的特殊情况。即若X→Y，则X→→Y。这是因为当X→Y时，对X的每一个值x，Y有一个确定的值y与之对应，所以X→→Y
  4）若X→→Y，X→→Z，则X→→YZ
  5）若X→→Y，X→→Z，则X→→Y∩Z
  6）若X→→Y，X→→Z，则X→→Y－Z，X→→Z－Y。

多值依赖与函数依赖

- 区别
  - 函数依赖规定某些元组不能出现在关系中，也称为相等产生依赖
  - 多值依赖要求某种形式的其它元组必须在关系中，称为元组产生依赖
- 有效性范围
  - X→Y的有效性仅决定于X、Y属性集上的值，它在任何属性集W（XY ⊆ W ⊆ U）上都成立。

    若X→Y在R(U)上成立，则对于任何Y′ ⊆ Y，均有X→Y ′ 成立。

多值依赖与函数依赖

- 有效性范围
  - X→→Y的有效性与属性集范围有关
  - X→→Y在属性集W（XY ⊆ W ⊆ U）上成立，但在U上不一定成立
  - X→→Y在U上成立 ⇒ X→→Y在属性集W（XY ⊆ W ⊆ U）上成立

嵌入型多值依赖

- 若函数依赖X→Y在R(U)上成立，则对于任何Y‘ ⊆Y均有X→Y’成立。而多值依赖X→→Y若在R(U)上成立，我们却不能断言对于任何Y‘ ⊆ Y有X→→Y’成立。因此有:
- 定义4.13：设关系模式R（U），X和Y是属性集U的子集，W是U的真子集，并且XY⊆W。如果MVD X→→Y在模式R上不成立，但在模式W上成立，则称X→→Y为R上的嵌入型多值依赖（Embedded MVD，简记为eMVD），用符号$(X\rightarrow\rightarrow Y)_W$表示

4.3 关系模式分解

- 我们可以通过把一个”大的”关系模式分解成多个”小的”关系模式，以解决插入异常、删除异常和更新异常等问题
- 实际上，关系模式的分解，不仅仅是属性集合的分解，也是对关系模式上的函数依赖集，以及关系模式的当前值的分解的具体表现
- 关系模式的分解并不是随意的，必须保证原来的关系模式的语义性质和信息不被丢失，即要求分解应当既”无损联接”又”保持函数依赖”

关系模式分解的定义

- 定义4.14:设R(U)为关系模式，则称:

  ρ ={R1(U1),R2(U2),…,Rk(Uk)} (其中U=$\bigcup_{i=1}^{k} U_i$,且对于任意的1≤i,j≤k没有Ui⊆Uj)为R的一个分解。

- 定义4.15:函数依赖集F在属性集Ui(⊆U)上的投影定义为:

$$\prod_{U_i}(F)=\{X \rightarrow Y \mid X \rightarrow Y \in F^{+} \wedge XY \subseteq U_i\}$$

- R称为泛关系模式，R对应的当前值称为泛关系
- 数据库对应的当前值σ 称为数据库实例，它是由数据库模式中的每一个关系模式的当前值组成， 即

  σ = {r1,r2…rk}

泛关系模式

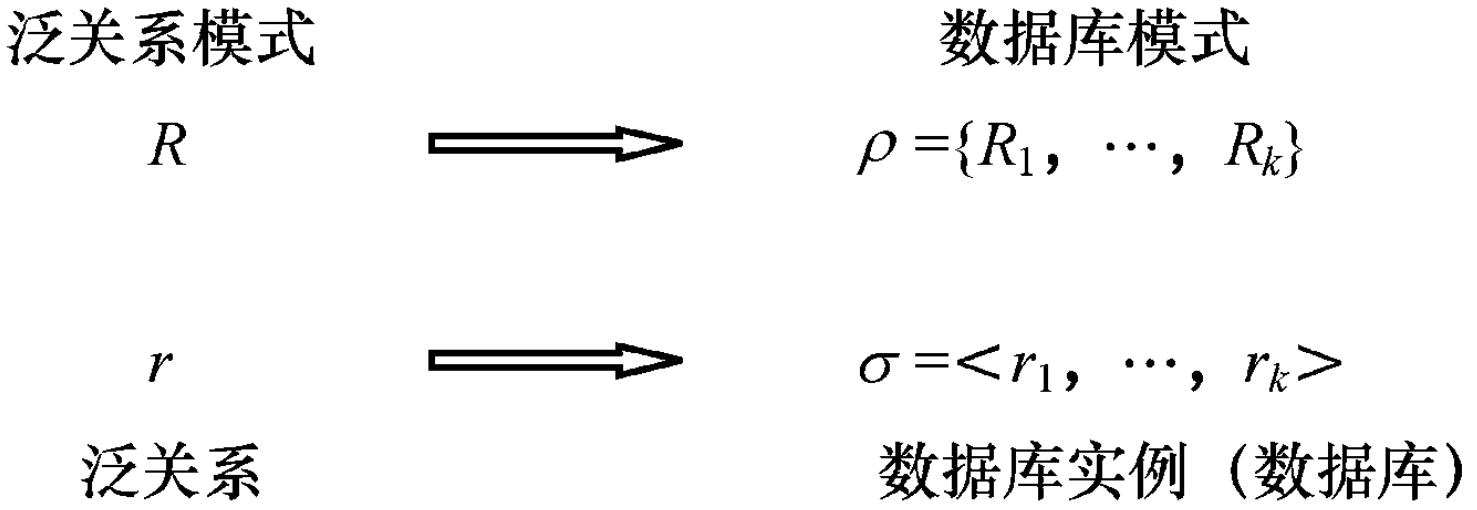


图4.5　模式分解示意图

无损联接分解

- 定义4.16:设R是一个关系模式，F是R上的一个FD集,ρ = { R_1，…，R_k } 为R的一个分解,如果对R中满足F的每一个关系r，都有

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \cdots \bowtie \pi_{R_k}(r)$$

- 则称分解ρ相对于F是“无损联接分解”(lossless join decomposition)，简称“无损分解”，否则，称ρ为”有损联接分解”(loss join decomposition),简称“有损分解”。

无损联接分解：例

- 例

r	A	B	C
	1	1	1
	1	2	1

（a）

r_1	A	B
	1	1
	1	2

（b）

r_2	A	C
	1	1

（c）

图　未丢失信息的分解

r	A	B	C
	1	1	4
	1	2	3

（a）

r_1	A	B
	1	1
	1	2

（b）

r_2	A	C
	1	4
	1	3

（c）

$r_1 \bowtie r_2$	A	B	C
	1	1	4
	1	1	3
	1	2	4
	1	2	3

（d）

图　丢失信息的分解

投影连接运算

- 定义4.17：设ρ = { R_1，…，R_k } 为R的一个分解，r是R上的任意一个具体关系，则r对于ρ 的投影连接运算定义为：

$$m_\rho(r) = > < \prod_{i=1}^{k} {}_{ui}(r)$$

- 引理4.4：设ρ = { R_1，…，R_k } 是R的一个分解，r是R的任一关系，r_i=π $_{Ri}$（r）（1≤i≤k），那么有下列性质：
  - ① $r \subseteq m_\rho(r)$；
  - ② 若s= $m_\rho(r)$，则π $_{Ri}$（s）=r_i；
  - ③ $m_\rho(m_\rho(r)) = m_\rho(r)$，这个性质称为幂等性（ Idempotent）

无损联接分解

- 由引理4.4可得到下面的结论：
- 如果 r≠ mρ（r），则ρ 不是无损分解。
- 由引理4.4(1)可知，r经过分解后再连接，如果ρ 不是无损分解，则所得到的结果的元组数总比原来的r多，即所谓“元组增加，信息丢失(有损)！”
- 由引理4.4(2)可知，对r进行mρ（r)变换后得到的s可满足条件：

$$\Pi_{Ri}(s) = ri$$

- 但必须注意：如果ri不是r的投影，而是独立的关系，则一般而言，连接后的关系s不再满足条件$\Pi_{Ri}(s) = ri$，而是$\Pi_{Ri}(s) \subseteq ri$

无损联接分解

- 例:设关系模式R（ABC）分解成ρ = { AB，BC }，（a）和（b）分别是模式AB和BC上的值r_1和r_2，（c）是$r_1 \bowtie r_2$的值。显然$\pi_{BC}(r_1 \bowtie r_2) \neq r_2$。这里$r_2$中元组（$b_2c_2$）就是一个悬挂元组，由于它的存在，使得$r_1$和$r_2$不存在泛关系r

r_1	A	B
	a_1	b_1

（a）关系r_1

r_2	B	C
	b_1	c_1
	b_2	c_2

（b）关系r_2

$r_1 \bowtie r_2$	A	B	C
	a_1	b_1	c_1

（c）$r_1 \bowtie r_2$

无损分解的判别方法

1. 构造一张k行n列的表格，每列对应一个属性A_j（$1 \leqslant j \leqslant n$），每行对应一个模式$R_i$（$1 \leqslant i \leqslant k$）。如果$A_j$在$R_i$中，那么在表格的第$i$行第$j$列处填上符号$a_j$，否则填上$b_{ij}$
2. 把表格看成模式R的一个关系，反复检查F中每个FD在表格中是否成立，若不成立，则修改表格中的值。修改方法如下：对于F中一个FD X→Y，如果表格中有两行在X值上相等，在Y值上不相等，那么把这两行在Y值上也改成相等的值。如果Y值中有一个是a_j，那么另一个也改成a_j；如果没有a_j，那么用其中一个b_{ij}替换另一个值（尽量把下标ij改成较小的数）。一直到表格不能修改为止（这个过程称为chase过程）
3. 若修改的最后一张表格中有一行是全a，即$a_1a_2\cdots a_n$，那么称ρ 相对于F是无损分解，否则称损失分解

无损分解的判别算法

- 算法4.3：检验一个分解是否无损连接分解
- 输入：关系模式R(A1,…,An);R上的函数依赖集F；
  R的一个分解ρ ={R1,R2,…,Rk}；
- 输出：　ρ 是否无损连接分解
- 方法：　（1）建立一个n列、k行的符号表M ：

	A1	A2	Aj	An
R1	…	…	…	…
R2	…	…	…	…
	…	…	…	…
Ri			M[i,j]	…
Rk	…	…	…	…

无损分解的判别算法

- 符号表M各元素的值由下面的规则确定：

$$M[i,j]=\begin{cases}a_j \text{若} A_j \in U_i \\ bij \text{若} A_j \notin U_i\end{cases}$$

- 用F中的每一函数依赖X→Y对M反复进行下列检查和处理：检查X中的属性所对应的列，找出在X上取值相等的行，如果找到两个(或多个)行在X上取值相等，就将对应行中Y中属性所对应的符号改为一致，即如果其中之一为aj，则将其他符号也改为aj；如果全部符号都是“b”符号，则将它们改为同样的“b”符号
- 如此进行下去，直到发现M中某一行变为：a1,a2,……,an，则说明ρ 是无损连接分解；
- 否则，一直进行到M不再改变为止，则说明ρ 不是无损连接分解

无损分解的判别算法

- 例：R(A,B,C,D,E)

  F={A→C, B→C, C→D, DE→C, CE→A}

- 判断:

  p＝{R1(A,D),R2(A,B),R3(B,E),R4(C,D,E),

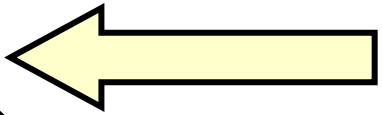
  R5(A,E)}

- 是否是无损连接的分解?

无损分解的判别算法

1、构造初始表格

	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

无损分解的判别算法

	A	B	C	D
AB	a_1	a_2	b_{13}	b_{14}
BC	b_{21}	a_2	a_3	b_{24}
CD	b_{31}	b_{32}	a_3	a_4

（a） 初始表格

	A	B	C	D
AB	a_1	a_2	b_{13}	b_{14}
BC	a_1	a_2	a_3	a_4
CD	b_{31}	b_{32}	a_3	a_4

（a） 修改后的表格

图4.12 算法4.3的运用示意图（一）

	A	B	C	D
AB	a_1	a_2	b_{13}	b_{14}
BC	b_{21}	a_2	a_3	b_{24}
CD	b_{31}	b_{32}	a_3	a_4

（a） 初始表格

	A	B	C	D
AB	b_1	a_2	b_{13}	b_{14}
BC	b_1	a_2	a_3	a_4
CD	b_{31}	b_{32}	a_3	a_4

（a） 修改后的表格

图4.13 算法4.3的运用示意图（二）

无损分解的判别算法

- 对于只有2个关系模式的分解可以通过下面的定理判断其是否为无损联接分解
- 定理4.4：设ρ = { R_1，R_2 } 是关系模式R的一个分解，F是R上成立的FD集，那么分解ρ 相对于F是无损分解的充分必要条件是（$R_1 \cap R_2$）→（$R_1 - R_2$）或（$R_1 \cap R_2$）→（$R_2 - R_1$）

- 定理4.5：如果FD：X→Y在模式R上成立，且X∩Y=φ，则ρ ={U－Y，XY }是R的无损分解

保持函数依赖的分解

- 保持关系模式分解等价的另一个重要条件是关系模式的函数依赖集在分解后仍在数据库模式中保持不变
- 即关系模式R 到ρ={R1,R2…Rk}的分解，应使函数依赖集F,被F在这些Ri上的投影蕴涵

保持函数依赖的分解

- 定义4.18：设ρ = { R1，…，Rk }是R的一个分解，F是R上的FD集，如果有：

$$\cup \pi_{R_i}(F) \models F$$

则称分解ρ 保持函数依赖集F(dependency-preserving decomposition)

保持函数依赖的分解

- 下面介绍检验分解是否保持函数依赖的算法
- 检验分解是否保持函数依赖实质上就是检验 $\bigcup_{i=1}^{k} U_i(F)$ 是否覆盖F
- 算法4.4：检验分解是否保持函数依赖。
- 输入：ρ ={R1,R2,…,Rk}，函数依赖集F
- 输出：ρ 是否保持F

- 方法：令G= $\bigcup_{i=1}^{k} U_i(F)$
- 为检验G是否覆盖F，可对F中的每一函数依赖X→Y进行下列检验：
- 首先计算 X_G^+，然后检查Y是否被包含在 X_G^+ 中

保持函数依赖的分解

- 下面是不必求出G而计算 X_G^+的算法：

```
Z := X;
repeat
   for i=1 to k do
      Z:=Z∪((Z ∩ Ui)F+ ∩ Ui)
until Z不再变化;
```

- 如果Y被包含在 X_G^+中，则X→Y $\in G^+$
- 如果F中的所有函数依赖经检验都属于 G^+ 则ρ保持依赖，否则不保持依赖

保持函数依赖的分解：举例

- 例4.6：R(ABCD) F={A→B,B→C,C→D,D→A}
  试判别ρ ={R1(AB),R2(BC),R3(CD)}是否保持依赖F
- 解：显然，ρ 可以保持 A→B,B→C,C→D
- 下面检验ρ 是否保持D→A?
- 令G= $\underset{i=1}{\overset{k}{Y}} X_{Ui}(F)$
- ∵ D_G^+ =DCBA 其中包含A
- ∴ ρ 保持依赖F

保持函数依赖的分解：举例

- 设关系模式R={CITY, ST, ZIP}满足函数依赖：

  (CITY, ST) → ZIP

  ZIP →CITY

  分解成ρ＝ {R1(ST, ZIP), R2(CITY, ZIP)}

- 检查该分解是否具有保持函数依赖性：

  ∵ $\prod_{R1}$=ø（平凡的函数依赖）

  $\prod_{R2}$={ZIP →CITY}

  $\prod_{R1} \cup \prod_{R2}$={ZIP →CITY}

- ∴ ρ不具有保持函数依赖性

模式分解与模式等价问题

- 本节讨论的关系模式分解的两个特性实际上涉及到两个数据库模式的等价问题，这种等价包括数据等价和依赖等价两个方面
- 数据等价是指两个数据库实例应表示同样的信息内容，用“无损分解”衡量
- 如果是无损分解，那么对泛关系反复的投影和联接都不会丢失信息
- 依赖等价是指两个数据库模式应有相同的依赖集闭包
- 在依赖集闭包相等情况下，数据的语义是不会出差错的。
- 违反数据等价或依赖等价的分解很难说是一个好的模式设计

模式分解与模式等价问题

- 例4.7：关系模式R（ABC），ρ = { AB，AC } 是R的一个分解
- 试分析分别在F_1= { A→B }，F_2= { A→C，B→C }，F_3= { B→A }，F_4= { C→B，B→A } 情况下，检验ρ 是否具有无损分解和保持FD的分解特性
- 解：
  ①相对于F1 = { A→B }，分解ρ 是无损分解且保持FD的分解
  ②相对于F2 = { A→C，B→C }，分解ρ 是无损分解，但不保持FD集。因为B→C丢失了
  ③相对于F3 = { B→A }，分解ρ 是损失分解但保持FD集的分解
  ④相对于F4 = { C→B，B→A }，分解ρ 是损失分解且不保持FD集的分解(丢失了C→B)

4.4 关系模式的规范化

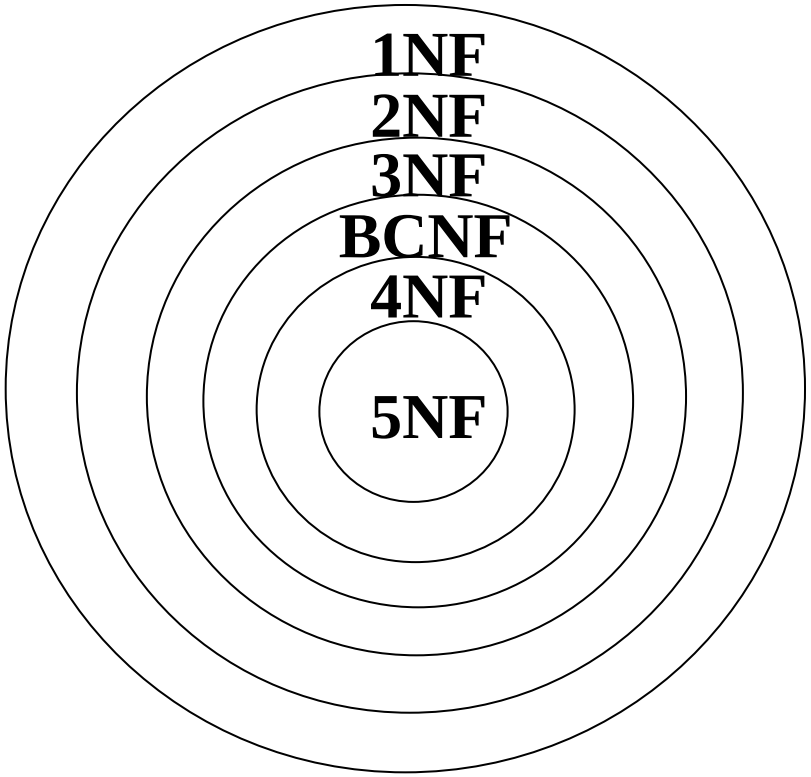
- 为了使数据库设计的理论和方法走向完备，人们研究了关系规范化理论，以指导我们设计规范化的数据库模式
- 在进行关系模式设计时，首先面临的一个问题是如何评价一个关系模式的“好”与“坏”，即用什么标准衡量？
- 这个标准就是模式的范式(Normal Forms，简记为NF)
- 而使得关系模式达到一定范式要求，就是关系模式的规范化

范式

- 关系数据库中的关系模式要满足一定的规范化要求，满足不同程度规范化要求的关系模式的类，称为不同的范式(Normal Forms，简记为NF)
- 范式的种类与数据依赖有着直接的联系，基于FD的范式有1NF、2NF、3NF、BCNF等多种
- 满足最低要求的关系模式称为第一范式，简称1NF。在第一范式中满足进一步要求的为第二范式，其余以此类推
- R为第几范式就写成R∈xNF

各种范式之间的关系

- 按属性间依赖情况来区分，关系规范化的程度为1NF, 2NF, 3NF, BCNF, 4NF 和 5NF等
  各种范式之间的关系如图所示：



规范化的过程

- 1NF是关系模式的基础；2NF已成为历史，一般不再提及；在数据库设计中最常用的是3NF和BCNF
- 为了叙述的方便，我们按照1NF、2NF、3NF、BCNF顺序来介绍

第一范式（1NF）

- 定义4.19：如果关系模式R的每个关系r的属性值都是不可分的原子值，则称R是第一范式（first normal form，简记为1NF）的模式
- 满足1NF的关系称为规范化的关系，否则称为非规范化的关系
- 关系数据库研究的关系都是规范化的关系。
- 例如关系模式R（NAME，ADDRESS，PHONE），如果一个人有两个电话号码（PHONE），那么在关系中至少要出现两个元组，以便存储这两个号码
- 1NF是关系模式应具备的最起码的条件

第一范式（1NF）

- 分量是否需要再分，与具体应用有关,如果用到值的一部分，则需要进一步分割,例如:

姓名	生日
王军	68.7.10
张立	69.7.10
李明	80.3.28

姓名	年	月日
王军	68	7.10
张立	69	7.10
李明	80	3.28

- 如果只是查询出生日期，则它满足1NF
- 如果查询两人生日是否相同，则只比较月、日，需要将生日分解，就不满足1NF
- 如果比较两人的生肖呢？

键(Key)

- 定义4.20：设K为R<U，F>中的属性或属性组合，若 $K \rightarrow U \in F^+$，则称K为R的一个超键
- 定义4.21：如果一个超键K的任何真子集都不再是超键，则称K为R的一个候选键(Candidate key)
- 候选键有时也简称为键
- 若关系模式中有多个候选键，则选定其中的一个为主键(Primary key)
- 定义4.22：包含在任何一个候选键中的属性，称为主属性(Prime attribute)或键属性(Key attribute)。不包含在任何候选键中的属性，则称为非主属性(Nonprime attribute)或非键属性(Non-key attribute)

键(Key)

- 最简单的情况，单个属性是候选键。最极端的情况下，整个属性组是候选键，并称为全键(All-key)

- 例：关系模式R(P,W,A),属性P表示演奏者，W表示作品,A表示听众。假设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。听众也可以欣赏不同演奏者的不同作品，这个关系模式的候选键为(P,W,A)，即All-key

求关系模式上全部候选键

- 求关系模式上的全部候选键的方法
- 输入：关系模式R(U，F)，其中F是最小覆盖
- 输出：R上的所有候选键
- 步骤(1)：将R的全部属性分为四类，分别是C1：不在任何函数依赖中出现的属性；C2：仅在函数依赖决定子中出现的属性；C3：仅在函数依赖右部出现的属性；C4：在函数依赖左部右部均有出现的属性。注意，任何候选键中必然会包括C1、C2中的属性

求关系模式上全部候选键

- 步骤(2)：若$(C1 \cup C2)^{+}=U$ 或者 $C4=\Phi$，则$C1 \cup C2$即为惟一的候选键；否则，逐一将C4中的属性加入$C1 \cup C2$并计算其闭包，若其闭包为U，则$C1 \cup C2$与该属性构成关系上的一个候选键，重复该过程直至找出所有的候选键

求关系模式上全部候选键

- 例4.8：求R(A，B，C，D)，F＝{A→B，B→C，BD→A}的所有候选键：
- 解：C1＝Φ，C2＝{D}，C3＝{C}，C4＝{A，B}，$(C1 \cup C2)^{+}$＝{D}，所以C1∪C2不是候选键，依次将C4中的A、B加入C1∪C2中计算其闭包可得：$(AD)^{+}$＝ADBC，$(BD)^{+}$＝BDAC，所以可知R的候选键为AD和BD

第二范式（2NF）

- 定义4.23：如果关系模式R中的所有非主属性都完全函数依赖于所有CK，则称R满足2NF，表示为R∈2NF
- 2NF是在1NF的基础上消除非主属性对于所有候选键的部分函数依赖
- 如果数据库模式中每个关系模式都是2NF，则称数据库模式为2NF的数据库模式

第二范式（2NF）

- 例4.14 :设关系模式R（S#，C#，GRADE，TNAME，TADDR）的属性分别表示学生学号、选修课程的编号、成绩、任课教师姓名和教师地址等意义。（S#，C#）是R的候选键
- R上有两个FD：（S#，C#）→（TNAME，TADDR）和C#→（TNAME，TADDR），由于前一个FD是部分依赖，因此R不是2NF模式。此时R的关系就会出现冗余和异常现象。譬如某一门课程有100个学生选修，那么在关系中就会存在100个元组，因而教师的姓名和地址就会重复100次。
- 如果把R分解成R1（C#，TNAME，TADDR）和R2（S#，C#，GRADE）后，部分依赖（S#，C#）→（TNAME，TADDR）就消失了。R1和R2都是2NF模式

第三范式（3NF）

- 定义4.24：如果关系模式R中的非主属性既不部分函数依赖也不传递函数依赖于R上的所有候选键，则称R满足3NF，表示为R∈3NF
- 3NF是在2NF的基础上消除非主属性对于键的传递依赖
- 如果数据库模式中每个关系模式都是3NF，则称其为3NF的数据库模式

第三范式（3NF）

- **例6.15** 在例6.14中，R2是2NF模式，而且也已是3NF模式。但R1（C#，TNAME，TADDR）是2NF模式，却不是3NF模式。如果R1中存在函数依赖C#→TNAME和TNAME→TADDR，那么C#→TADDR就是一个传递依赖，即R1不是3NF模式。此时R1的关系中也会出现冗余和异常操作。譬如一个教师开设五门课程，那么关系中就会出现五个元组，教师的地址就会重复五次
- 如果把R2分解成R21（TNAME，TADDR）和R22（C#，TNAME）后，C#→TADDR就不会出现在R21和R22中。这样R21和R22都是3NF模式

第三范式（3NF）

- 推论1:如果R是3NF模式，那么R也是2NF模式

- 推论2:设关系模式R，当R上每一个FD X→A满足下列三个条件之一：A∈X（即X→A是一个平凡的FD）；X是R的超键；A是主属性。则关系模式R就是3NF模式

第三范式（3NF）

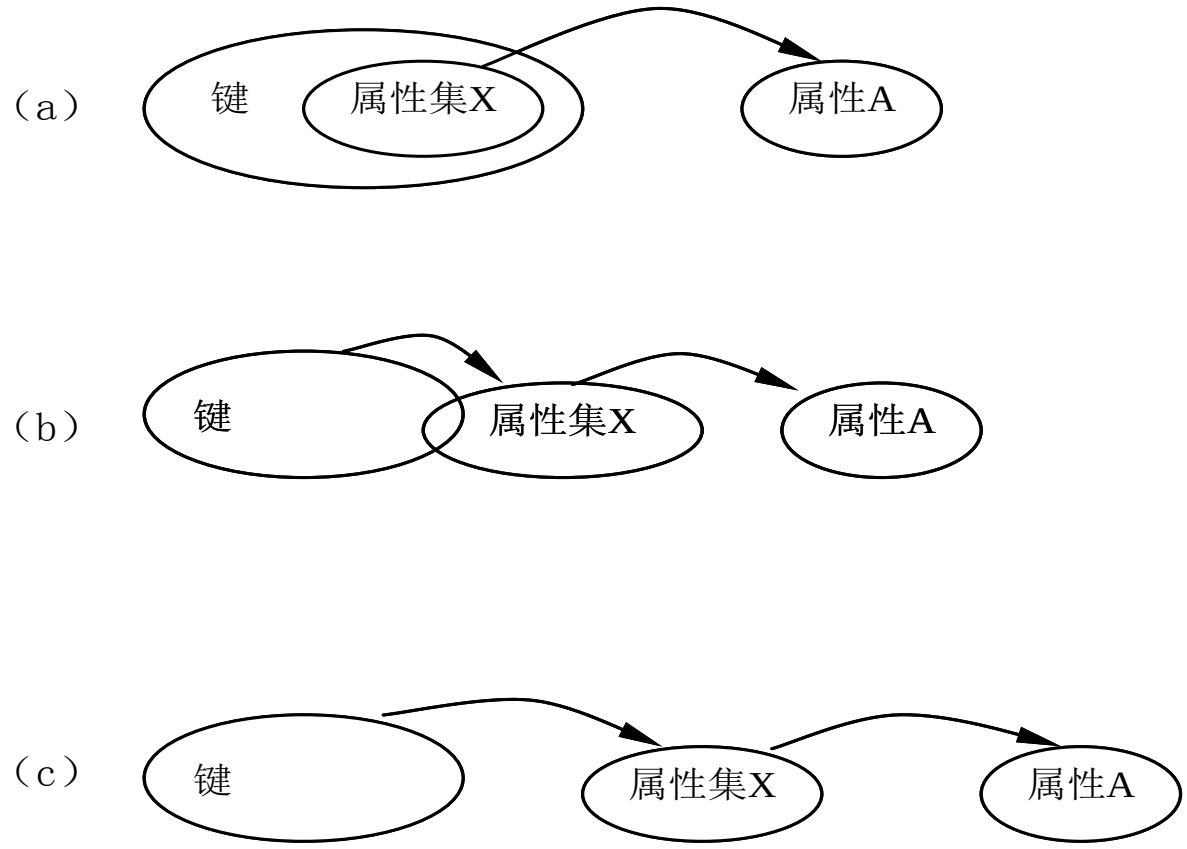


图6.15　违反3NF的传递依赖的三种情况

BCNF（Boyce–Codd NF）

- 定义4.25：若对于R上的任何非平凡函数依赖X→Y都有X必包含R的某个候选键，则称R满足BCNF，表示为R∈BCNF
- 定义4.25’：如果关系模式R中的每个属性都不部分或传递依赖于R的候选键，那么称R是BCNF的模式

BCNF（Boyce–Codd NF）

- 推论3：如果R是BCNF模式，那么R也是3NF模式
- BCNF是在3NF的基础上消除主属性对于键的部分依赖和传递依赖
- 如果数据库模式中每个关系模式都是BCNF，则称为BCNF的数据库模式。

BCNF（Boyce–Codd NF）

- 例：关系模式Teaching(Cname，Class，Tname)
- 假定每门课程由多名教师承担多个班级的教学任务，同时规定每名老师仅限承担一门课程的教学任务，但是可以同时给若干个班级带课，于是有(Cname，Class)→Tname，Tname→Cname
- Teaching 是3NF，但不是BCNF

第四范式（4NF）

- 定义4.26:设D是关系模式R上成立的FD和MVD集合。如果D中每个非平凡的MVD X→→Y的左部X都是R的超键，那么称R是4NF的模式
- 4NF就是限制关系模式的属性之间不允许有非平凡且非函数依赖的多值依赖
- 因为根据定义，对于每一个非平凡的多值依赖X→→Y，X都含有候选键，于是就有X→Y，所以4NF所允许的非平凡的多值依赖实际上是函数依赖

第四范式（4NF）

- 显然，如果一个关系模式是4NF， 则必为BCNF。但反之不然
- 多值依赖的“毛病”在于数据冗余太大。我们可以用投影分解的方法消去非平凡且非函数依赖的多值依赖
- 函数依赖和多值依赖是两种最重要的数据依赖
- 如果只考虑函数依赖，则属于BCNF的关系模式规范化程度为最高。如果考虑多值依赖，则属于4NF的关系模式规范化程度为最高

判断范式级别

- 给定关系模式和函数依赖集合，要求判断达到的最高范式。
- 方法：

1. 求出给定关系的候选键（可能不止一个）
2. 根据键，确定主属性和非主属性。
3. 判断是否满足第一范式(看属性的值域是否可以分解）
4. 判断是否满足第二范式(非主属性对键的部分函数依赖)
5. 判断是否满足第三范式(非主属性对键的传递函数依赖)
6. 判断是否满足BCNF范式(主属性对键的传递函数依赖)

判断范式级别：举例

例1:已知关系r（A,B,C,D,E）和F={AB→CE,E→AB,C→D},试判断该关系最高为第几范式?

解：

1、关系r的候选键为：AB和E。

因为:$(AB)^+$={ABCED}

$(E)^+$={EABCD}

AB可以决定关系的所有属性，E可以决定关系的所有属性，所以AB 和E是关系的候选键

2、主属性为：A、B、E，非主属性为：C、D

3、判断是否满足各个范式的要求：

- 因为r的所有的属性值域都是不可再分，所以r已经在1NF中了
- 该关系为2NF，因为C、D不存在对任何键的部分依赖
- 该关系不是3NF，因为存在非主属性对键的传递依赖C→D

判断范式级别：举例

例2:已知关系r（A,B,C,D,E,F）和F={A→B,C →DF,AC→E,D→F},该关系最高为第几范式?

解：

- 该关系的键为AC，因为$(AC)^+$={ACBDFE}
- 主属性为：A、C
- 非主属性为：B,D,E,F
- 该关系明显满足1NF，但是由于A→B,C →DF存在非主属性对键的部分函数依赖，所以该关系不是2NF
- 所以r满足的最高范式是1NF。

判断范式级别：举例

例3：已知关系r（A,B,C,D）和F={AB→D,AC→BD,B→C}，试求出其最高范式？

解：该关系的键是：AB和AC。

- 主属性是{ABC}，非主属性是{D}。
- D不部分依赖于任何键，所以该关系在2NF中。
- 该关系也在3NF中，因为只有一个非主属性，从而不可能存在两个不同非主属性间的传递依赖
- 该关系不在BCNF中，因为有B→C，而B不是r的键

分解成3NF模式集的算法

- 算法4.4：既无损连接又保持函数依赖的3NF分解算法
- 输入：关系模式$R_R(U_R, F_R)$
- 输出：$\rho = \{ R_0, R_1, \cdots, R_k, R_{ck}\}$

分解成3NF模式集的算法

•步骤：

(1)首先将F_R转换为等价的最小覆盖，记为F；

(2)将不在F中出现的属性单独构成关系模式$R_0(U_0)$，其余属性记为$U=U_R-U_0$，于是得到第一个分解 $=\{R_0, R\}$，其中R(U，F)；

(3)如果R中有某一个函数依赖包含U中的全部属性，则算法终止，输出即为ρ ；

分解成3NF模式集的算法

(4)否则，将F中的函数依赖按照左部相同原则进行分组，假设共分为k组，得到k个关系模式$R_i(U_i, F_i)$，U_i表示每组函数依赖所涉及的全部属性，F_i是相应的函数依赖集，将这k个关系模式加入ρ 中得到：ρ ＝{ R_0，R_1，…，R_k}；

分解成3NF模式集的算法

(5)经过上述步骤得到的每个R_i均是3NF，且因为分组是按照函数依赖进行的，不会丢失任何一个函数依赖，所以该过程是保持函数依赖的。为了保证该分解同时是无损连接的，需要再加入一个由任一候选键构成的关系模式R_{ck}，如果该关系模式已在前面的某个R_i中出现，则可以省略；

(6)输出 $\rho = \{ R_0, R_1, \cdots, R_k, R_{ck}\}$，算法结束。

分解成3NF模式集的算法

- 例4.6：与学生有关的关系模式R(S#，Sname，Province，City，Street，Zipcode)，各属性分别表示学号、姓名、来自省份、城市、街道及邮政编码
- R上有函数依赖集F＝{S#→Sname，S#→Province，S#→City，S#→Street，S#→Zipcode，(Province，City，Street)→Zipcode，Zipcode→Province，Zipcode→City}，试将其分解为3NF形式且保持无损连接及函数依赖

分解成3NF模式集的算法

- 解：
- (1)首先计算F的最小覆盖Fmin，因S#→Zipcode冗余，所以：

  Fmin＝{S#→Sname，S#→Province，S#→City，S#→Street，(Province，City，Street)→Zipcode，Zipcode→Province，Zipcode→City}；

分解成3NF模式集的算法

- (2)由于R的所有属性均在Fmin中出现，所以直接将函数依赖分组构成新的关系模式， ρ＝{R1(S#, Sname, Province, City, Street), R2(Province, City, Street, Zipcode), R3(Zipcode, Province, City)}，因R3是R2的子集，故将其略去，得到 ρ＝{R1(S#, Sname, Province, City, Street), R2(Province, City, Street, Zipcode)};

分解成3NF模式集的算法

- (3)关系R上的候选键只有一个S#，因其也是R1的子集，所以不再加入ρ中，于是满足要求的最终关系模式集合仍为:

  ρ＝{R1(S#, Sname, Province, City, Street),

  R2(Province, City, Street, Zipcode)}

分解成BCNF模式集的算法

- 算法4.5：保证无损连接的BCNF分解算法
- 输入：关系模式R(U，F)
- 输出：　ρ＝{ R1，…，Rk}
- 步骤：

(1)令ρ＝{R(U，F)}；

(2)如果ρ中的每个子关系模式均是BCNF，则算法终止，输出即为　ρ；

分解成BCNF模式集的算法

(3)否则， ρ中必有某个子关系模式Ri(Ui，Fi)不满足BCNF，根据BCNF的定义，至少有一个$X \rightarrow A \in F^{+}_{i}$，且X不包含Ri的任一候选键，将Ri分解为两个关系模式Ri1、Ri2，且Ri1＝{XA，Fi1}，Ri2＝{Ui－A，Fi2}，用Ri1、Ri2取代Ri，返回(2)继续执行

分解成BCNF模式集的算法:举例

- 已知关系模式R(CTHRSG),其中
- C:表示课程　　T:表示教师　　H:表示时间
- R:表示教室　　S:表示学生　　G:表示成绩
- C →T:每门课程仅有一名教师讲授
- 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，一个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

键为HS

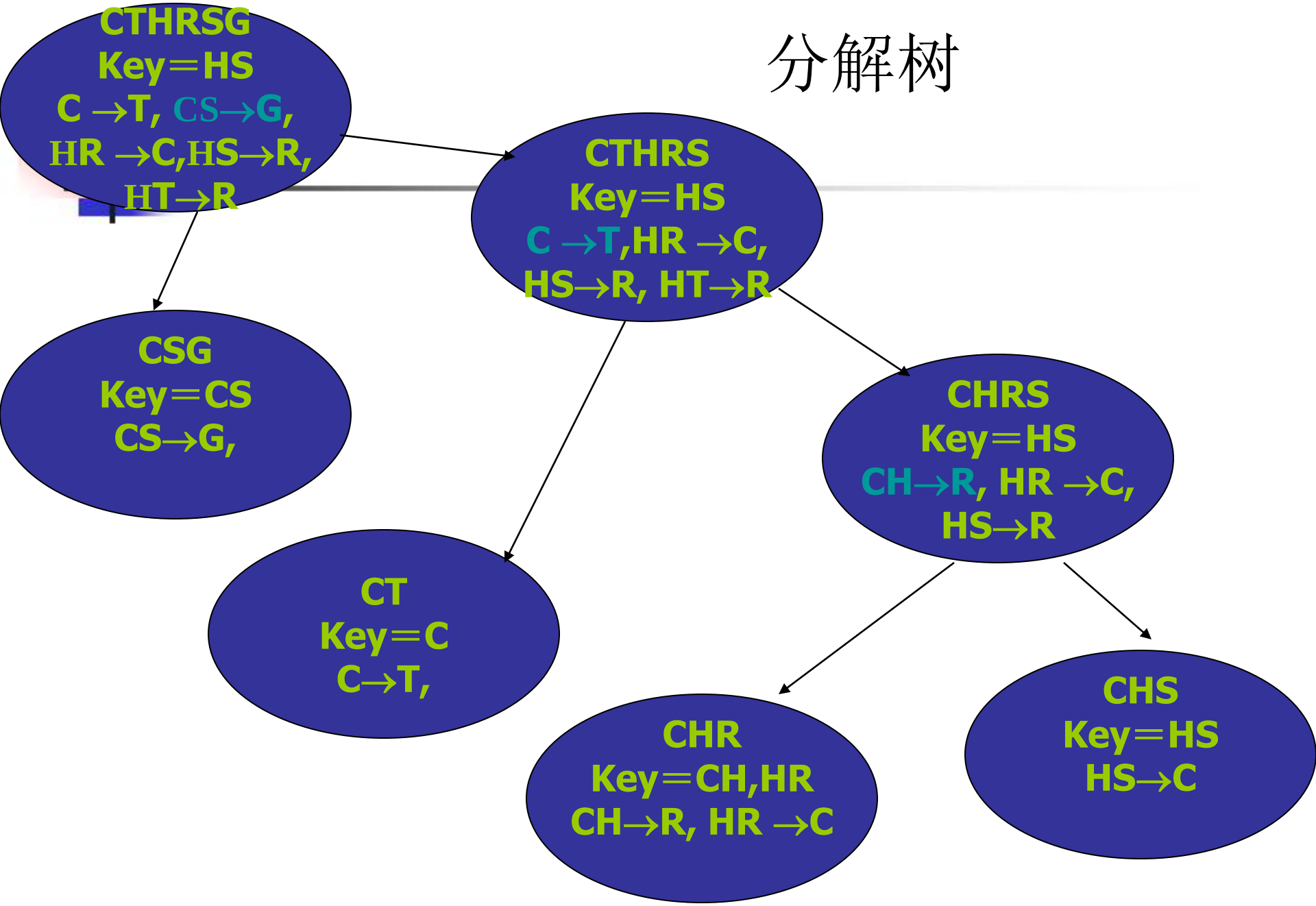
分解成BCNF模式集的算法:举例

- CTHRSG经过三次分解，分解为CSG,CT,CHR,CHS模式。这四个关系模式分别描述了：
  - 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

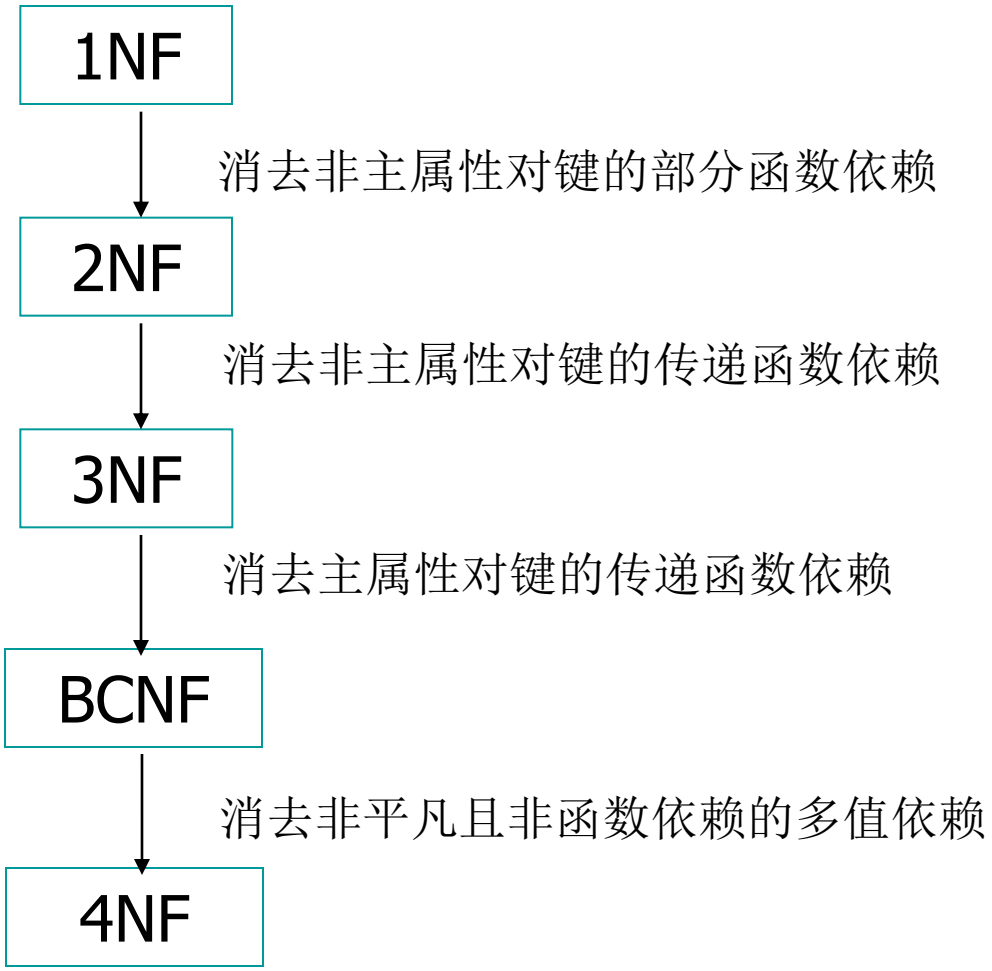
键为HS

- 这些模式都是BCNF，而且这些分解是无损分解。
- 但是保持函数依赖吗？

分解树


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

规范化过程总结


1NF
消去非主属性对键的部分函数依赖
2NF
消去非主属性对键的传递函数依赖
3NF
消去主属性对键的传递函数依赖
BCNF
消去非平凡且非函数依赖的多值依赖
4NF

关系模式设计的原则

- 我们在进行关系数据库设计时，应作权衡，尽可能使数据库模式保持最好的特性
- 一般尽可能设计成BCNF模式集，如果设计成BCNF模式集时达不到保持FD的特点，那么只能降低要求，设计成3NF模式集，以求达到保持FD和无损分解的特点
- 模式分解并不单指把泛关系模式分解成数据库模式，也可以把数据库模式转换成另一个数据库模式，分解和转换的关键是要“等价”地分解
- 一个好的模式设计应符合三条原则：表达性、分离性和最小冗余性