规范化的基本思想

- 消除不合适的数据依赖
- 各关系模式达到某种程度的“分离”
- 采用“一事一地”的模式设计原则

让一个关系描述一个概念、一个实体或者实体间的一种联系。若多于一个概念就把它“分离”出去。

规范化实质是概念的单一化

（一张表只解决一个问题）

6.2.9 规范化小结

- 防止另一个极端：不能说规范化程度越高的关系模式就越好。

在设计数据库模式结构时，必须对现实世界的实际情况和用户应用需求作进一步分析，确定一个合适的、能够反映现实世界的模式。

（上面的规范化步骤可以在其中任何一步终止）

第六章 关系数据理论

6.1 问题的提出

6.2 规范化

6.3 数据依赖的公理系统

6.4 模式的分解

数据依赖的公理系统是对关系模式进行分解、进行规范化处理的理论基础。

为了表述简洁和推理方便，对有关记号使用做如下约定：

⑴ 如果声明**X**、**Y**等是属性子集，则将**X**∪**Y**简记为**XY**。

⑵ 如果声明**A**、**B**等是属性，则将集合｛**A**，**B**｝简记为**AB**。

⑶ 如果**X**是属性集，**A**是属性，则将**X**∪｛**A**｝简记为**XA**或**AX**。

以上是两个对象的情形，对于多个对象也做类似约定。

⑷ 关系模式简记为三元组**R(U,F)**，其中**U**为模式的属性集合，**F**为模式的函数依赖集合。

6.3　数据依赖的公理系统

函数依赖的逻辑蕴涵：

当讨论函数依赖时，经常需要从一些已知的函数依赖中去判断另外一些函数依赖是否成立。

例如，如果A→B和B→C在某个关系中成立，记作F={A→B，B→C}，那么A→C在该关系中是否成立的问题就称为逻辑蕴涵问题。

6.3 数据依赖的公理系统

- 逻辑蕴含

定义6.11 对于满足一组函数依赖 F 的关系模式R <U，F>，其任何一个关系r，若函数依赖$X \rightarrow Y$都成立, 则称

F**逻辑蕴含**$X \rightarrow Y$

Armstrong公理系统

- 一套推理规则，是模式分解算法的理论基础
- 用途
  - 求给定关系模式的**码**
  - 从一组函数依赖求得蕴含的**函数依赖**

Armstrong公理系统

对关系模式$R<U，F>$来说有以下的推理规则：

- A1.**自反律**（Reflexivity）：
  若$Y \subseteq X \subseteq U$，则$X \rightarrow Y$为$F$所蕴含。
- A2.**增广律**（Augmentation）：若$X \rightarrow Y$为F所蕴含，且$Z \subseteq U$，则$XZ \rightarrow YZ$为F所蕴含。
- A3.**传递律**（Transitivity）：若$X \rightarrow Y$及$Y \rightarrow Z$为F所蕴含，则$X \rightarrow Z$为F所蕴含。

函数依赖公理

Armstrong公理系统的推论：

- A4（合成规则）若$X \rightarrow Y$，$X \rightarrow Z$，则$X \rightarrow YZ$。
- A5（分解规则）若$X \rightarrow YZ$，则 $X \rightarrow Y$，$X \rightarrow Z$。
- A6（ 伪传递规则）若$X \rightarrow Y$，$WY \rightarrow Z$，则$XW \rightarrow Z$。

引理：

$X \rightarrow A_1A_2 \ldots A_k$成立的充分必要条件是$X \rightarrow A_i$成立$(i=1, 2, \ldots, k)$。

函数依赖公理

Armstrong公理是有效的，完备的。

- **有效性：**

由F出发根据Armstrong公理推理出的每一个函数依赖一定在F^+**中**。

- **完备性：**

F^+中每一个函数依赖，必定可以由F出发根据Armstrong公理推导出来。

属性闭包及其运算

- **F^+(F的闭包):**

定义：所有被F逻辑蕴涵的函数依赖集称为**F的闭包**(Closure)，记作F^+。一般情况下，$F \subseteq F^+$，如果$F=F^+$，则称F是函数依赖的完备集。

对于U上的一个函数依赖(FD)X→Y，如何判定它是属于F^+，即如何判定是否F逻辑蕴含X→Y呢？一个自然的思路就是将F^+计算出来，然后看X→Y是否在集合F^+ 之中 。

规定：若X为U的子集，则X→Φ属于F^+。

设有关系模式R(U,F)，其中U＝ABC，F＝｛A→B，B→C｝

求F的闭包 F^+

A→Φ	AB→Φ	AC→Φ	ABC→Φ	B→Φ	C→Φ
A→A	AB→A	AC→A	ABC→A	B→B	C→C
A→B	AB→B	AC→B	ABC→B	B→C	Φ→Φ
A→C	AB→C	AC→C	ABC→C	B→BC	
A→AB	AB→AB	AC→AB	ABC→AB	BC→Φ	
A→AC	AB→AC	AC→AC	ABC→AC	BC→B	
A→BC	AB→BC	AC→BC	ABC→BC	BC→C	
A→ABC	AB→ABC	AC→ABC	ABC→ABC	BC→BC	

有没有更简便的方法来判断 X→Y ∈ F+

属性闭包及其计算

引理6.2：

设关系模式R(U，F)，U为R的属性集合，F为其函数依赖集，$X\subseteq U$且$Y\subseteq U$，则从F推出$X\rightarrow Y$的充分必要条件是$Y\subseteq X^+$。

判断$X\rightarrow Y$是否包含在F^+中，只要判断$Y\subseteq X^+$是否成立。这样就把计算F^+的问题简化为计算X^+的问题。

属性闭包的定义： 设关系模式R(U，F)，$U=\{A_1，A_2，\dots，A_n\}$，F为其函数依赖集，X为U中的属性集，则称所有用Armstrong公理从F推出的函数依赖$X\rightarrow A_i$中A_i的属性集合为X的属性闭包，记作X^+(或X_F^+)读作X关于函数依赖集F的闭包。

$X_F^+=\{A \mid X\rightarrow A$能由$F$根据Armstrong公理导出$\}$。

属性闭包及其计算

- 算法6.1

输入：R(U,F)， U的子集为X

输出： X关于F的属性闭包X^+

方法：计算X(i)(i=0,1,…)。

(1) X(0)=X

(2) X(i+1)=X(i)A

在F中寻找尚未用过的函数依赖:$Y_j \rightarrow Z_j$(j=0,1,…,k)，其左边（Y_j）是X(i)的子集

A为Z_j中，但X(i)中未出现过的属性集合，若无A则转(4)

(3)判断是否有X(i+1)=X(i)，若有则转(4); 否则(2)。

(4) 输出X(i)，即为X^+。

对于(3)的计算停止，以下四种方法是等价的

- **X(i+1)=X(i)**
- **发现X(i)=U**
- **F中函数依赖的右边属性中全部已在X(i)中出现**
- **在F中未用过的函数依赖的左边属性已没有X(i)的子集**

属性闭包及其计算

例题：

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)+。

解：(1)由属性闭包算法，初始X(0)+ =AB。

(2) 逐一扫描集合 F 中各个函数依赖，找左部为A，B或AB的函数依赖。由AB→C，B→D，于是X(1)+=AB∪CD=ABCD。继续。

(3)继续扫描集合 F 还未使用的函数依赖，有C→E，AC→B，其左部为ABCD子集，于是 X(2)+ = ABCD∪BE=ABCDE。

(4)因为X(2)+ ==U，停止扫描。所以(AB)+=ABCDE=U。

属性闭包及其计算

例题：

设有关系模式R(A，B，C，D，E，I)，其中，F={A→D，AB→C，BI→C，ED→I，C→E}，计算(AC)+。

解：

(1) X(0)+ =AC；

(2) 在F中找出左部是AC子集的函数依赖：有A→D，C→E；得：X(1)+ =X(0)+∪D∪E=ACDE；继续。

(3) 在F中还未用过的函数依赖中，找出左部是ACDE子集的函数依赖：有ED→I；X(2)+=X(1)+∪I=ACEDI； 继续。

(4) X(2)+≠ X(1)+，但是F中未用过的函数依赖的左边属性已没有X(2)+的子集，所以，可停止计算，X(2)+即为(AC)+

(AC)+=ACDEI。

求解关系模式的候选码

对于给定的关系模式R(U，F)，可将其属性分成4类。

（1）L类：仅出现在F的函数依赖左部的属性。

（2）R类：仅出现在F的函数依赖右部的属性。

（3）N类：在F的函数依赖左右两边均未出现的属性。

（4）LR类：在F的函数依赖左右两边均出现的属性。

求解关系模式的候选码

$X \to U(U=A_1, A_2, \ldots, A_n$，关系模式R的所有属性)成立的充分必要条件是$X \to A_i$成立$(i=1, 2, \ldots, n)$。

从F推出$X \to A_i$的充分必要条件是$A_i \subseteq X^+$，若$X^+=U$，则$X \to U$，且X的任一真子集$(X')^+ \neq U$，即X必为码。

结论：

① L类属性和N类属性必包含于任何候选码中；

② R类属性必不包含于任何候选码中；

③ LR类属性不能确定是否在候选码中。

求解关系模式的候选码

定理 对于给定的关系模式R及其函数依赖集F，如果X(X∈ R)是L类属性，则X必为R的任一候选码的成员。

求解关系模式的候选码

定理　对于给定的关系模式R及其函数依赖集F，如果X(X∈R)是R类属性，则X不在任何候选码中。

定理　对于给定的关系模式R及其函数依赖集F，如果X是R的N类属性，则X必包含在R的任一候选码中。

推论　对于给定的关系模式R及其函数依赖集F，如果X是R的N类和L类组成的属性集，且X^+包含了R的全部属性，则X是R的唯一候选码。

求解关系模式的候选码的算法

(1)依照函数依赖集F将R中的所有属性分为L类、R类、LR类和N类属性，令X为L、N类属性的集合，Y为LR类属性集合；若X为空，转(3)

(2)若X_F^+=U，则X为R的唯一候选码，算法结束；否则继续；

(3)若$X_F^+ \neq$ U,令Y'＝Y，逐一取Y'中的单一属性A，令Y'＝Y'－{A}，若$(XA)_F^+$=U，则XA为候选码，直至Y'为空；

(4)依次取Y中的任意两个、三个……属性Z与X组成属性组，若XZ不包含已求得的候选码，求其关于F的闭包$(XZ)_F^+$，若$(XZ)_F^+$=U，则XZ为候选码。直到取完Y中的所有属性为止，算法结束。

求解关系模式的候选码

例：设有R(A，B，C，D)，其函数依赖集F={D→B，B→D，AD→B，AC→D}，求R的所有候选码。

解：A、C属性是L类属性，AC必是R的候选码成员；

又因为(AC)+=ACBD，AC是R的唯一候选码。

求解关系模式的候选码

例：设有R(A，B，C，D，E，P)，R的函数依赖集F={A→D，E→D，D→B，BC→D，DC→A}，求R的所有候选码。

解：C、E是L类属性，故C、E必在R的任何候选码中

P为N类属性，故P也必在R的任何候选码中。

又因为(CEP)+=ABCDEP，CEP是R的唯一候选码。

求解关系模式的候选码

例：设R(A,B,C,D,E)，F={A→BC, CD→E, B→D, E→A}，求R的所有候选码。

解: (1) R中无L、N类属性，ABCDE均为LR类属性；

(2) 在{ABCDE}中取单个属性测试：取A，则A_F+ =ABCDE=U，A为候选码；

分别取B、C、D，三个单属性的闭包均不等于全属性集U；

取E，则E_F+=ABCDE=U，E为候选码；

(3) 在{BCD}中任取两个属性判定:

$(BC)_F$ +=BCDEA=U，BC为候选码；

$(BD)_F$ +不等于全属性集U，BD不是候选码；

$(CD)_F$+=CDEAB=U，CD为候选码；

(4) BCD包含已求得的候选码BC，不是码，结束。

故关系R的候选码为：A，E，BC，CD。

求解关系模式的候选码

例：设有关系模式R(A，B，C)，其函数依赖集F={A→B，B→C}，则R最高达到第几范式？

解：A是L类属性，故A必在R的候选码中

因为(A)+=ABC=U，A是R的唯一候选码。

因为函数依赖集F中有： A→B，B→C 即非主属性C对码A存在传递函数依赖，但不存在部分函数依赖，故R最高达到2NF。

函数依赖公理

●最小函数依赖集

函数依赖集的闭包是给定FD所蕴涵的全部的属性间的函数依赖关系。换一个角度，怎样用最少的函数依赖来表达全部的属性间的依赖关系？这就是最小函数依赖集讨论的内容。

- 定义：假设在关系模式R<U, F>上有两个函数依赖集F和G。如果$F^+=G^+$，则称F和G是等价的，或称F与G相互覆盖。
- 定理：$F^+=G^+$的充分必要条件是 $F\subseteq G^+$且$G\subseteq F^+$。
- 定理：任一函数依赖集总可以为一右部都为单属性的函数依赖集所覆盖

函数依赖公理

定义：如果函数依赖集 F 满足下列条件，则称F为一个极小函数依赖集。又称为最小依赖集或最小覆盖（正则覆盖） F_m：

(1) F 中任一函数依赖的右部仅含有一个属性；

(2) F中不存在这样的函数依赖X→A，使得F与F－{X→A}等价；

F中每个函数依赖 都不可以被去掉/不是多余的

(3) F中不存在这样的函数依赖X→A，X有真子集Z使得 (F－{X→A})∪{Z→A}与F等价。

F中每个函数依赖左部 X 已经是最小集，不能换成其真子集

F的极小函数依赖集F_m不一定是唯一的。

函数依赖公理

正则覆盖的概念：

将 FD 的 F 减少到最小，找出最小的、最简单的 FD 集合 F_m，而且保证它能够蕴涵全部 $f \in F$。

函数依赖公理

- 最小函数依赖集求解算法-形式化描述：

（1）对F中每一函数依赖$X \to Y$，若$Y= A_1A_2\ldots A_m (m \ge 2)$，则用$\{ X \to A_i \mid i=1,\ldots,m \}$替换$X \to Y$；

（2） 对F中每一函数依赖$X \to A$，若$X= B_1B_2\ldots B_n$，逐一考察B_i，若$A \in (X-B_i)_F^+$，则用$(X-B_i) \to A$替换$X \to A$；B_i称为无关属性；

（3） 对于F中的每一函数依赖$X \to A$，若$A \in X_{F-\{X \to A\}}^+$，则从F中去掉$X \to A$。

最小函数依赖集求解算法

最小函数依赖集求解算法：

输入：一个函数依赖集F。

输出：F的一个等价最小依赖集F_m。

(1)右边属性单一化。

应用分解规则，使F的每个函数依赖的右部属性都为单属性。

(2) 依次去除F的每个函数依赖左部多余的属性。

设XY→A是F的任一函数依赖，在F中求出X的闭包X_F^+。如果X_F^+包含了A，则Y为多余属性，该函数依赖就替换为X→A。

(3) 依次去除多余的函数依赖。

设X→A是F的任一函数依赖，在F- {X→A}中求出X的闭包$X_{F-\{X\to A\}}^+$。如果$X_{F-\{X\to A\}}^+$包含了A，则X→A为多余的函数依赖，应该去除；否则，不能去除。**若去除，令F= F-{X→A}，从（3）重新开始计算。**

最小函数依赖集求解

例：设关系模式R(ABCDE) 上的函数依赖集F={A→BC, BCD→E, B→D, A→D, E→A}，求F的最小函数依赖集。

（1） 对每个函数依赖右部属性分离，得：

F_1={A→B, A→C, BCD→E, B→D, A→D, E→A}

（2）去掉左部冗余属性 考察BCD→E,

∵ $(BC)_F^+$=BCDEA，包含E，BCD→E中的D为冗余属性，以BC→E取代。而BC→E左部已无冗余属性 ，因为$(B)_F^+$= BD，$(C)_F^+$=C 不含E。得：

F_2＝{A→B, A→C, BC→E, B→D, A→D, E→A}

（3） 去掉多余函数依赖

∵ 由A→B, B→D可以得到A→D，故A→D是多余的，去掉。

去掉后再依次考察剩下的函数依赖，按照求最小函数依赖集算法依次检验，再无多余的函数依赖。

∴ Fm＝{A→B, A→C, BC→E, B→D, E→A}

最小函数依赖集求解

一个给定的函数依赖集F的最小函数依赖集不是唯一的。

- 原因：在求解过程中由于考察FD的次序的不同会产生不同的结果，但一个函数依赖集F的所有最小函数依赖集是等价的（都等价于F）。

例：$F = \{ A \to B, B \to A, B \to C, A \to C, C \to A \}$

F_{m1}、F_{m2}都是F的最小依赖集：

$F_{m1} = \{ A \to B, B \to C, C \to A \}$

$F_{m2} = \{ A \to B, B \to A, A \to C, C \to A \}$

第六章 关系数据理论

6.1 问题的提出

6.2 规范化

6.3 数据依赖的公理系统

6.4 模式的分解

6.4 模式的分解

- 范式是通过数据依赖(函数依赖和多值依赖)对关系模式进行规范，范式间关系：1NF ⊃ 2NF ⊃ 3NF ⊃ BCNF ⊃ 4NF。
- 规范化的过程就是对关系模式进行分解，使其达到更高一级的范式，来减少和避免更新异常和数据冗余。
- 满足BCNF的关系模式可以在函数依赖的范畴内解决更新异常和数据冗余。

6.4 模式分解

●模式分解的定义

定义： 关系模式R<U, F>的一个分解

$$\rho=\{ R_1<U_1,F_1>, \ R_2<U_2,F_2>, \ \ldots, \ R_n<U_n,F_n>\}$$

其中：① $U=\bigcup_{i=1}^{n} Ui$，并且不存在 $U_i \subseteq U_j$，$1 \le i, j \le n, i \neq j$；

② F_i 为与函数依赖集 $\{ X\to Y \mid X\to Y \in F^+ \wedge XY \subseteq U_i \}$ 等价的一个函数依赖集，称为 F在 U_i 上的投影, 记为 $\prod_{Ri}(F)$ 。

6.4 模式分解

- **模式分解应考虑的问题：**
  - 分解不能丢失信息

如学生关系S(Sno, Sname, Ssex, Sage, Sdept)，它的一个实例 *r* 如下：

sno	sname	ssex	sage	sdept
95001	李勇	男	20	CS
95002	刘晨	女	19	IS
95003	王敏	女	18	MA

其中的每条记录都是一个学生实体的描述，如(95001,李勇，男，20, CS)，表示：学号为95001的同学名叫李勇，男性，今年20岁，在CS系学习。

如果将该关系模式分解为$\rho=\{S_1(Sno), S_2(Sname), S_3(Ssex), S_4(Sdept)\}$，则任一关系实例都被分解为一些离散的值，不再表示原关系中的意义，即分解丢失了信息。

6.4 模式分解

➢分解应该能够被还原

模式的分解是为了更好地存储，在使用中通过自然连接还原为分解前的关系模式，如下表为一银行的存款记录：

R:

客户	帐号	存款
李勇	**a01**	**80**
李勇	**a02**	**100**

R_1:

客户	帐号
李勇	**a01**
李勇	**a02**

R_2:

客户	存款
李勇	**80**
李勇	**100**

将其分解为右上图的两个关系R_1和 R_2:

还原后的关系模式如右图：

多出了两条记录！(×)

$R_1 \bowtie R_2$:

客户	帐号	存款
李勇	a01	80
李勇	a01	100
李勇	a02	80
李勇	a02	100

➢分解应保持函数依赖

函数依赖是属性间自身具有的性质，通常的查询与此有关，故应保持。

关系模式分解的标准

- 分解具有无损连接性
- 分解要保持函数依赖
- 分解既要保持函数依赖，又要具有无损连接性

无损连接

- 模式分解的无损连接性(Lossless join)

引理6.4：设关系模式R<U, F>的一个分解ρ={ R_1<U_1,F_1>，R_2<U_2,F_2>，…，R_k<U_k,F_k>}，r是R的一个关系实例，$r_i=\pi_{Ri}(r)$为关系r在模式R_i上的投影，则：（证明见P196）

(1) $r \subseteq m_\rho(r) = \bowtie_{i=1}^{k} r_i = \bowtie_{i=1}^{k} \pi_{Ri}(r)$

(2) 若 $s = m_\rho(r)$，则$\pi_{Ri}(s) = r_i$

(3) $m_\rho(m_\rho(r)) = m_\rho(r)$

此定理描述了分解之后的关系与原关系的联系。

定义6.18：ρ={ R_1<U_1,F_1>，R_2<U_2,F_2>，…，R_k<U_k,F_k>}是关系模式R<U, F>的一个分解，若对于R< U, F >的任一关系r均有$r = m_\rho(r)$ 成立，则称**分解ρ具有无损连接性**。简称**ρ为无损分解**。

无损连接(续)

例：设有关系模式R（A, B, C），$\rho=\{R_1,R_2\}$为它的一个分解，其中R_1=AB，R_2=BC, r是R的一个关系，$r_1=\pi_{R_1}(r)$，$r_2=\pi_{R_2}(r)$。请问该例中ρ是否是一个无损分解？

r

A	B	C
a1	b1	c1
a2	b1	c2
a1	b1	c2

$r_1=\pi_{R_1}(r)$

A	B
a1	b1
a2	b1

$r_2=\pi_{R_2}(r)$

B	C
b1	c1
b1	c2

$r \subseteq s=m_\rho(r)=r_1 \bowtie r_2$

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2

$S_1=\pi_{R_1}(S)=r_1$

A	B
a1	b1
a2	b1

$S_2=\pi_{R_2}(S)=r_2$

B	C
b1	c1
b1	c2

s_1

A	B
a1	b1
a2	b1

s_2

B	C
b1	c1
b1	c2

$m_\rho(m_\rho(r)) = s_1 \bowtie s_2 = m_\rho(r)$

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2

$r \neq m_\rho(r)$, 所以分解ρ不是一个无损分解

无损连接

例： 学生成绩登记表SCG：

Sno	Sname	Ssex	Sage	Sdept	Cno	Cname	Grade
s95001	李勇	男	20	CS	c1	数据库	92
s95001	李勇	男	20	CS	c2	高等数学	65
s95001	李勇	男	20	CS	c4	操作系统	88
s95002	刘晨	女	19	IS	c2	高等数学	90
…	…	…	…	…	…	……	…

分解为如下三个表(分别为S、C、SC)：

Sno	Sname	Ssex	Sage	Sdept
s95001	李勇	男	20	CS
s95002	刘晨	女	19	IS
…	…	…	…	…

Cno	Cname
c1	数据库
c2	高等数学
c4	操作系统
…	……

Sno	Cno	Grade
s95001	c1	92
s95001	c2	65
s95001	c4	88
s95002	c2	90
…	…	…

SCG＝(S ⋈ C ⋈ SC)，为无损分解。

无损连接性判定算法

- **无损连接性的判定算法：**

设ρ={ R_1， R_2， …， R_k}是关系模式R<U, F>的一个分解，U= {A_1, …, A_n}， F={ FD_1,FD_2, …, FD_m }，(假设F为最小函数依赖集，若不是，求之，可提高求解效率), 并设FD_i为$X_i \rightarrow A_i$。

(1) 建立一张k行n列的表，每一行对应一个关系模式，每一列对应一个属性，若属性A_j属于R_i，则在i行j列交叉处填上a_j，否则填上b_{ij}；

例： 已知R<U, F>， U= { A, B, C, D, E }， F={ AB→C , D→E, C→D}， R的一个分解ρ={R_1(A,B,C), R_2(C,D), R_3(D,E)}。判定分解ρ是否为无损连接的分解。

解： (1) 构造初始表：

R_i \ U	A	B	C	D	E
(ABC)	a_1	a_2	a_3	b_{14}	b_{15}
(CD)	b_{21}	b_{22}	a_3	a_4	b_{25}
(DE)	b_{31}	b_{32}	b_{33}	a_4	a_5

无损连接判定算法

(2) 逐个检查F中的每一个函数依赖X→Y，在X的分量中寻找相同的行，然后将这些行中Y的分量改为相同的符号，如果这些行中Y分量中有a_j，则将这些分量中的b_{ij}改为a_j，若这些行中Y分量没有a_j，则将这些行所有Y分量改成b_{ij}（i为这些分量中行号最小值）(称为一趟Chase过程)

注：**若某个b_{ij}被更改，那么该表中的j列中凡是b_{ij}的符号均应做相应修改**（不管它是否为位于本步骤开始时，在X的分量中找相同的行时找到的那些行内）

(例)：……，(F={ AB→C, D→E , C→D }，……。

解：(1) 构造初始表；

a_5

R_i \ U	A	B	C	D	E
(ABC)	a_1	a_2	a_3	b_{14}	b_{15}
(CD)	b_{21}	b_{22}	a_3	a_4	b_{25}
(DE)	b_{31}	b_{32}	b_{33}	a_4	a_5

(2) 第一趟Chase：

① AB→C：无AB列上取值相同的行，不做修改；

② D→E：第二、三行在D上的取值均为a_4，第三行E列出现a_5，将这两行在E上的取值修改为a_5；

无损连接判定算法

(例)：……，(F={ AB→C, D→E , C→D }，……。

(2) 第一趟Chase：

① ……

② ……

R_i \ U	A	B	C	D	E
(ABC)	a_1	a_2	a_3	b_{14}	b_{15}
(CD)	b_{21}	b_{22}	a_3	a_4	a_5
(DE)	b_{31}	b_{32}	b_{33}	a_4	a_5

a_4

③ C→D：第一、二行在C上的取值为a_3，将这两行在D上的值改为a_4；

无损连接判定算法

(3) 如果在一趟Chase中的某次更改之后出现形如$a_1,a_2,\ldots,a_n$的行，算法结束，分解ρ具有无损连接性；否则，继续下一趟Chase过程，直到一趟Chase之后表无任何变化时算法结束 (此时未出现形如$a_1,a_2,\ldots,a_n$的行)，分解ρ不具有无损连接性。

(例)： ……，(F={ AB→C, D→E , C→D }，……。

(2) 第一趟Chase后的结果：

R_i \ U	A	B	C	D	E
(ABC)	a_1	a_2	a_3	a_4	b_{15}
(CD)	b_{21}	b_{22}	a_3	a_4	a_5
(DE)	b_{31}	b_{32}	b_{33}	a_4	a_5

a_5

(3) 第二趟Chase：

① AB→C：无改变；

② D→E：第一、二、三行在D上的取值为a_4，将它们在E上的取值改为a_5；

(4)表第一行出现a_1,a_2,a_3,a_4,a_5，故**分解ρ为无损分解**。

无损连接性

定理6.5： 关系模式R<U, F>的一个分解ρ = { $R_1<U_1,F_1>$, $R_2<U_2,F_2>$ }具有无损连接性的充分必要条件是：$U_1 \cap U_2 \rightarrow U_1 - U_2 \in F^+$ 或 $U_1 \cap U_2 \rightarrow U_2 - U_1 \in F^+$。

用于一分为二的模式分解无损连接性的判定

例： 学生关系S (Sno, Sname, Ssex, Sdept, Mname)， Mname 系主任

分解为 S(Sno, Sname, Ssex, Sdept) 和 D(Sdept, Mname),

$U_D \cap U_S$= Sdept

$U_D - U_S$ = Mname

Sdept $\rightarrow$ Mname为原关系中的函数依赖，

此分解为无损连接的分解。

函数依赖保持性

定义6.19：若 $F^+ = (\bigcup_{i=1}^{n} F_i)^+$，则R<U, F>的分解$\rho$={ R_1<U_1, F_1>，R_2<U_2, F_2>，…，R_k<U_k, F_k> } **保持函数依赖**。

定义：设R<U, F>，Z是R的一个属性集合，则称Z所涉及到的F^+中所有函数依赖为F在Z上的投影，记为$\prod_Z$（F）={X→Y| X→Y∈F^+且XY ⊆ Z} 注：X→Y∈ F^+等价于Y ⊆ X^+

设关系模式R <U, F>的分解ρ={ R_1，R_2，…，R_k}，若F等价于$\prod_{R1}$（F）∪ $\prod_{R2}$（F）∪ ··· ∪ $\prod_{Rn}$（F），则称ρ具有依赖保持性

函数依赖保持性

例：将$R=(ABCD, \{A\rightarrow B, C\rightarrow D\})$分解为关于$U_1=AB$，$U_2=CD$两个关系，求$R_1$、$R_2$，并检验分解的无损连接性和分解的函数依赖保持性。

令 $F_1=\prod_{R1}(F)=\{A\rightarrow B\}$,

令$F_2=\prod_{R2}(F)=\{C\rightarrow D\}$,

则 $R_1=(AB, \{A\rightarrow B\})$

$R_2=(CD, \{C\rightarrow D\})$

$U_1\cap U_2=AB\cap CD=\Phi$

$U_1-U_2=AB$　　$U_2-U_1=CD$

$\Phi\not\rightarrow AB$

$\Phi\not\rightarrow CD$

所以ρ不是无损分解；

$F_1\cup F_2=\{A\rightarrow B, C\rightarrow D\}$

$\equiv F$

所以ρ具有函数依赖保持性。

函数依赖保持性

例：将R=(ABC，{AB→C，C→A})分解为R_1<U_1,F_1>, R_2 <U_2,F_2>两个关系，其中U_1=BC，U_2=AC，该分解是否具有函数依赖保持性。

令 $F_1=\prod_{R1}(F)=\Phi$，

令$F_2=\prod_{R2}(F)=\{C\to A\}$，

则 $R_1=(BC，\Phi)$

$R_2=(AC，\{C\to A\})$

$F_1 \cup F_2 \neq F$

所以ρ不具有函数依赖保持性

无损连接和保持函数依赖关系

- 一个无损连接分解不一定具有依赖保持性
- 一个依赖保持性分解不一定具有无损连接性

模式分解算法

- 算法6.3 转换为3NF的保持函数依赖的分解(合成法)
- 算法6.4 转换为3NF既有无损连接性又保持函数依赖的分解
- 算法6.5 转换为BCNF且具有无损连接性的算法(分解法)

算法6.3 合成法

- 转换为3NF的保持函数依赖的分解

输入：关系模式R和R的最小函数依赖集F_m

输出：R<U,F>的一个分解ρ={ R_0，R_1，…，R_i}，**R_i为3NF，ρ保持函数依赖**。

(1) 如果R中某些属性（记为U_0）与F_m中所有依赖的左部与右部都无关，则将它们构成关系模式，记为R_0 <U_0,F_0> 。

(2)如果F_m中有一依赖X→A，且XA=U，则输出ρ=（R），算法终止。

(3)对F_m具有相同左部的原则分组，分为i组，每一组函数依赖所涉及的全部属性形成一个属性集U_k。若$U_k \subseteq U_j$，就去掉U_k($j \neq k$)。

(4) 停止分解，输出ρ={ R_1 <U_1,F_1>，…，R_k <U_k,F_k>} ∪ R_0 <U_0,F_0> 。

算法6. 3 合成法

例：将X(U=CTHRSG，F={CS→G，C→T, TH→R，HR→C，HS →R})分解为保持依赖性的3NF。

F_m=F={CS→G，C→T, TH→R， HR→C，HS →R}

按照算法6.3步骤

（1）不满足条件

（2）不满足条件

（3）X_1=CSG, X_2=CT, X_3=THR, X_4=HRC, X_5=HSR

（4）ρ={CSG， CT， THR， HRC， HSR}

算法6. 4

- 转换为3NF既有无损连接性又保持函数依赖的分解

输入：针对R<U, F>进行算法6.3（合成法）得到分解ρ ={ R_1 $<U_1,F_1>$，…，R_k $<U_k,F_k>$} $\cup R_0$ $<U_0,F_0>$。

输出： 3NF既有无损连接性又保持函数依赖的分解τ

(1) X是R<U, F>的码 。令 $\tau = \rho \cup \{R^*<X, F_x>\}$

(2)若某个 U_i , $X \subseteq U_i$ ，将$R^*<X, F_x>$从 τ中去掉；或者$U_i \subseteq X$，将$R^*<X, F_x>$从 τ中去掉

(3) τ为输出所求

算法6. 4

例：将X(U=CTHRSG，F={HT→R，C→T, HR→C，HS→R，CS →G})分解为既有无损连接性又保持函数依赖性的3NF的关系模式。

（1）X的唯一候选码为HS

（2）合成法后得到分解ρ={X_1（HTR）, X_2（CT）, X_3（HRC）, X_4（HSR）, X_5（CSG）}

（3） $\tau = \rho \cup$ {R*（HS）}

（4）X_4所涉及属性集U_4包含码HS，所以τ中去掉X*，最终分解τ={X_1（HTR）, X_2（CT）, X_3（HRC）, X_4（HSR）, X_5（CSG）}

算法6. 5 分解法

- **算法6.5 :**分解为BCNF且具有无损连接性的算法(分解法)

(1) 求F的最小函数依赖集F_{min}并替代F；

(2) 令ρ = { R<U, F> }；

(3) 如果ρ中各关系模式都属于BCNF，算法结束；

(4) 如果ρ中某个$R_i<U_i,F_i>\notin$BCNF，则必有一函数依赖$X\rightarrow A\in F_i^+(A\nsubseteq X)$，且X非$R_i$的码，对$R_i$分解为：$R_{i1}(XA)$和$R_{i2}(U_i-A)$，用{$R_{i1}$ ，R_{i2}}代替R_i，转(3)。

若R不满足BCNF，开始分解，初始R_i=R，U_i=U

注：因为关系模式中属性为有限个，算法会在有限次循环后结束。分解结果不一定唯一，因为与选定的X→A有关。

自顶向下，二叉分解树。

算法6.5 分解法

例：将X(U=CTRHSG，F={CS→G, HS →R, C→T，TH→R, HR→C})分解为一组BCNF的关系模式，要求分解具有无损连接性。

F_m=F={CS→G, HS →R, C→T，TH→R，HR→C}

第一遍循环：

①X的唯一候选码为HS，多个函数依赖不包含码，因此X不属于BCNF

②由CS→G，CS不包含候选码，分解X为X_1=CSG和X_2=CTRHS（注意去掉了G），并分别求得CSG和CTRHS上函数依赖集：

F_1={CS →G}　F_2={HS →R, C →T, TH→R，HR→C}；

③ F_{1m}= F_1，F_{2m}= F_2 （通过F_1, F_2计算的最小函数依赖集就是各自本身）

④ F_{1m}计算出X_1的唯一候选码为CS，因此X_1属于BCNF。F_{2m}计算出X_2的唯一候选码为HS，所以X_2不属于BCNF。

ρ_1={<u>CS</u>G, CTR<u>HS</u>}

算法6.5 分解法

第二遍循环：

X_2=（ CTR<u>HS</u> ，{HS →R, C →T, TH→R， HR→C}）

①对第一步中的X_2继续分解，由C→T，C不包含X_2的唯一候选码为HS，分解X_2（ CTRHS ）为CT和CRHS（注意去掉了T）

②分别求CT和CRHS上最小函数依赖集：{C →T} 和 {HS →R，HR→C}

③ CT（<u>C</u>T，{C →T} ）（属于BCNF）

CRHS（CR<u>HS</u>，{HS →R，HR→C}）（不属于BCNF）

所以，ρ_2={<u>CS</u>G, <u>C</u>T，CR<u>HS</u>}

算法6.5 分解法

第三遍循环：

CRHS (CR<u>HS</u>，{HS →R, HR→C})

① 对CRHS继续分解，由HR →C，HR不包含CRHS的唯一候选码HS，分解CRHS为CHR和RHS（注意去掉了C）

②分别求CHR和RHS上最小函数依赖集：{HR →C} 和 {HS →R}

③ CHR（C<u>HR</u>，{HR →C}）（属于BCNF）

RHS（R<u>HS</u>，{HS →R}）（属于BCNF）

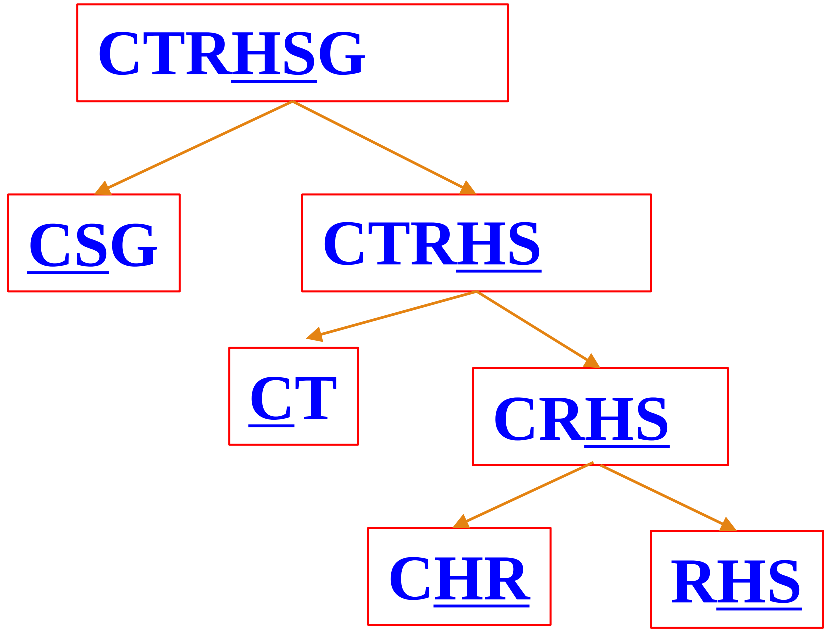
所以，ρ_3={<u>CSG</u>, <u>CT</u>, C<u>HR</u>, R<u>HS</u>}

$\rho=\rho_3$,每个子模式都达到BCNF，且分解具有无损连接性

分解法输出结果并不唯一，与选择分解的函数依赖次序有关，若第一次分解选择考虑C →T，则分解结果为？是否一致呢？

算法6.5 分解法

二叉树：



$$\rho_3=\{\underline{\mathrm{CS}}\mathrm{G},\ \underline{\mathrm{C}}\mathrm{T},\ \mathrm{C}\underline{\mathrm{HR}},\ \mathrm{R}\underline{\mathrm{HS}}\}$$

模式分解算法

- 若只要求分解保持函数依赖，那么模式分解总可以达到3NF，但不一定能够达到BCNF。
- 若要求分解既具有无损连接性，又保持函数依赖，则模式分解可以达到3NF，但不一定能够达到BCNF。

小结

- 规范化理论为数据库设计提供了理论的指南和工具
  - 也仅仅是指南和工具
- 并不是规范化程度越高，模式就越好
  - 必须结合应用环境和现实世界的具体情况合理地选择数据库模式

作业

- P203　第6题（不用交）