

第6章 数据库设计理论



5.3 数据库设计正确性分析

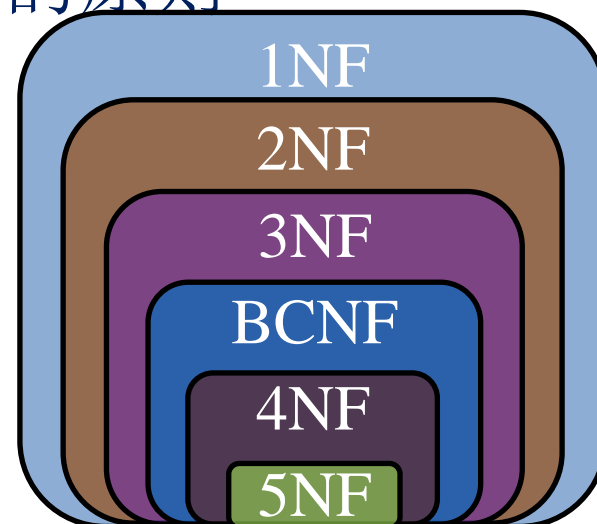
--规范的数据库设计

➤ 明确数据库表中的属性在取值方面存在的依存关系

- ❑ 函数依赖;
- ❑ 完全函数依赖与部分函数依赖;
- ❑ 传递函数依赖;

➤ 明确数据库设计过程中应遵循的原则

- ❑ 关系第1范式原则 (1NF)
- ❑ 关系第2范式原则 (2NF)
- ❑ 关系第3范式原则 (3NF)
- ❑ 关系Boyce-Codd范式原则 (BCNF)
- ❑ 4NF、5NF等



第6章 数据库设计理论

理论：概念、性质/公理/定理/推论、证明、算法等

6.1 函数依赖理论

—关于属性之间关系的理论

6.2 关系范式理论

—关于关系模式设计的规范化形式的理论

6.3 模式分解理论(关系模式规范化方法)

—关于大模式在分解为小模式过程中的理论

6.4 小结

5.3 数据库设计正确性分析

--不正确设计数据库所引发的问题

➤ 插入异常和删除异常的另外的例子

- ❑ 若删除张一，则级别4信息丢失
- ❑ 若无人的级别是7，则级别7的工资信息无法在表中反映
- ❑ 若有人级别未定，则不能插入该表中

职工表(职工, 级别, 工资)

职工	级别	工资
张一	4	800
张二	6	2000
张三	5	1000
张四	6	2000

级别	工资
7	3000

职工	级别	工资
钱三	(未定)	(未定)

5.3 数据库设计正确性分析

--问题的根源：
不良的函数依赖

职工表
(职工, 级别, 工资)

职工	级别	工资
张一	4	800
张二	6	1500
张三	5	1000
张四	6	1500

职工→级别
级别→工资

蕴含

职工→工资

工资依赖于职工

职工的工资级别表

职工	级别
张一	4
张二	6
张三	5
张四	6

职工→级别

工资不再依赖于职工

工资级别信息表

级别	工资
4	800
5	1000
6	1500
7	2000

级别→工资

分解

6.1 数据依赖理论

--函数依赖

➤ 函数依赖

关系中，两个属性(或属性组)X和Y，如果X上的值相等，则Y上的值一定相等，那么我们说X函数决定Y，或说Y函数依赖于X，记为： $X \rightarrow Y$ 。

例如：U = {学号，姓名，年龄，班号，班长，课号，成绩}

- ❑ 学号 \rightarrow {姓名，年龄}
- ❑ 班号 \rightarrow 班长
- ❑ {学号，课号} \rightarrow 成绩

6.1 数据依赖理论

--函数依赖

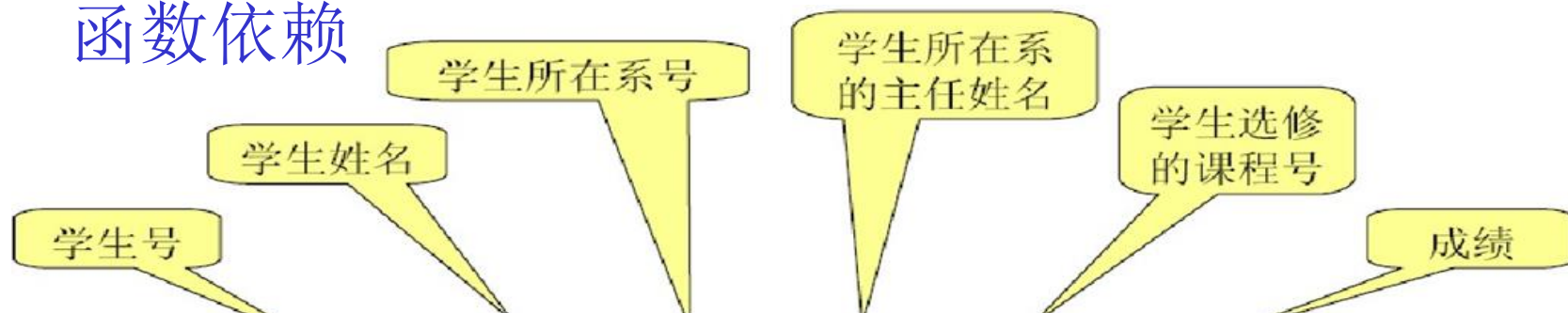
➤ 函数依赖需注意问题

- ❑ 函数依赖的分析取决于对问题领域的限定和分析。
 - ✓ 问题领域中，学生是没有重名的，则有“年龄”和“家庭住址”都函数依赖于“姓名”。
 - ✓ 而在另一个问题领域中，学生是有重名的，则上述函数依赖是不成立的。
- ❑ 函数依赖的正确分析取决于对问题领域内业务规则的正确理解。
- ❑ 当设计一个关系模式时，除给出属性全集外，还需给出属性全集上的所有数据依赖。

6.1 数据依赖理论

--函数依赖

- 学生的关系模式S (S#, SN, SD, DEAN, C#, G) 中存在的函数依赖



S#	SN	SD	DEAN	C#	G
S01	杨明	D01	思齐	C01	90
S02	李婉	D01	思齐	C01	87
S01	杨明	D01	思齐	C02	92
S03	刘海	D02	述圣	C01	95
S04	安然	D02	述圣	C02	78
S05	乐天	D03	省身	C01	82

6.1 数据依赖理论

--函数依赖

➤ 学生的关系模式S (S#, SN, SD, DEAN, C#, G) 中存在的函数依赖

S#	SN	SD	DEAN	C#	G
S01	杨明	D01	思齐	C01	90
S02	李婉	D01	思齐	C01	87
S01	杨明	D01	思齐	C02	92
S03	刘海	D02	述圣	C01	95
S04	安然	D02	述圣	C02	78
S05	乐天	D03	省身	C01	82

复合属性做为
决定因素

6.1 数据依赖理论

--函数依赖

➤ 学生的关系模式 $S(S\#, SN, SD, DEAN, C\#, G)$ 中存在的函数依赖

S#	SN	SD	DEAN	C#	G
S01	杨明	D01	思齐	C01	90
S02	李婉	D01	思齐	C01	87
S01	杨明	D01	思齐	C02	92
S03	刘海	D02	述圣	C01	95
S04	安然	D02	述圣	C02	78
S05	乐天	D03	省身	C01	82

复合属性做为 决定因素	单个属性作为 决定因素	假如学生 不能重名
$(S\#, C\#) \rightarrow SN$	$S\# \rightarrow DEAN$	$SN \rightarrow DEAN$
$(S\#, C\#) \rightarrow SD$	$S\# \rightarrow SN$	$SN \rightarrow S\#$
$(S\#, C\#) \rightarrow DEAN$	$S\# \rightarrow SD$	$SN \rightarrow SD$
$(S\#, C\#) \rightarrow G$	$SD \rightarrow DEAN$	

6.1 数据依赖理论

--函数依赖

➤ 函数依赖的分类

□ 部分依赖

设 X, Y 是关系 R 的两个属性集合, 存在 $X \rightarrow Y$, 若 X' 是 X 的真子集, 存在 $X' \rightarrow Y$, 则称 Y 部分函数依赖于 X 。

□ 完全依赖

设 X, Y 是关系 R 的两个属性集合, X' 是 X 的真子集, 存在 $X \rightarrow Y$, 但对每一个 X' 都有 $X' \nrightarrow Y$, 则称 Y 完全函数依赖于 X 。

□ 传递函数依赖

设 X, Y, Z 是关系 R 中互不相同的属性集合, 存在 $X \rightarrow Y (Y \nrightarrow X), Y \rightarrow Z$, 则称 Z 传递函数依赖于 X 。

6.1 数据依赖理论

--函数依赖

➤ 函数依赖的分类

▣ 平凡依赖

当关系中属性集合Y是属性集合X的子集时，存在函数依赖 $X \rightarrow Y$ ，即一组属性函数决定它的所有子集，这种函数依赖称为平凡函数依赖。

▣ 非平凡依赖

当关系中属性集合Y不是属性集合X的子集时，存在函数依赖 $X \rightarrow Y$ ，则称这种函数依赖为非平凡函数依赖。

6.1 数据依赖理论

--函数依赖

➤ 部分或完全函数依赖

在一个属性集合中，若属性(组) X 能够函数决定属性(组) Y ，但是， X 的任何一个子集(部分)都不能函数决定 Y ，则说 X 完全决定 Y ，或说， Y 完全函数依赖于 X ；否则说 Y 部分函数依赖于 X 。

6.1 数据依赖理论

--函数依赖

➤ 部分或完全函数依赖

□ 例如：学生(学号，课程号，成绩)

“学号，课程号”完全函数决定“成绩”

□ 再如：学生(学号，学生姓名，课程号，课程名，成绩)

“课程名”和“学生姓名”都部分函数依赖于“学号，课程号”

□ 再如：学生(学号，学生姓名，系号，系主任)

“系主任”部分函数依赖于“学号，系号”，

“姓名”部分函数依赖于“学号，系号”

6.1 数据依赖理论

--函数依赖

➤ 分析下列模式是否存在部分函数依赖

- ❑ 学生(学号, 学生姓名, 班级, 课程号, 课程名, 成绩, 任课教师, 教师职务)
- ❑ 员工(员工编码, 姓名, 出生日期, 联系方式, 最后学历, 毕业学校, 培训日期, 培训内容, 职务变动日期, 变动后职务)
- ❑ 图书(书号, 书名, 出版日期, 出版社, 书架号, 房间号)
- ❑ 客户(客户号, 客户名称, 类别, 联系电话, 欲购产品编码, 产品名称, 数量, 要货日期)

6.1 数据依赖理论

--函数依赖

➤ 传递函数依赖

在一个属性集合中，若属性(组)X能够函数决定属性(组)Y, 而Y又函数决定属性(组)Z, 但Y不能函数决定X, 而且Y不包含于X, 则说Z传递函数依赖于X。

▣ 例如：学生(学号, 姓名, 系号, 系主任)

“学号”函数决定“系号”，而“系号”函数决定“系主任”，则“系主任”传递依赖于“学号”

6.1 数据依赖理论

--函数依赖

➤ 候选键与非主属性

- 在一个属性集合中，能够完全决定所有属性的那些属性(组)被称为候选键
 - ✓ 一个属性集合可以拥有多个候选键
- 在一个属性集合中，包含在某一候选键中属性为主属性；而不包含在任何候选键中的属性，被称为非主属性。

下例中候选键是??非主属性是??

例如：学生(学号，年龄，家庭住址，课程号，成绩，任课教师，教师职务)

再如：邮编(城市名，街道名, 邮政编码)

6.1 数据依赖理论

--函数依赖

➤ 分析下列模式有无关于非主属性的传递函数依赖

- ❑ 商店(商店, 商品, 商品经营部, 商品经营部经理)
- ❑ 学生(学号, 学生姓名, 班级, 班主任, 课程号, 课程名, 成绩, 任课教师, 教师职务)
- ❑ 员工(员工编码, 姓名, 部门, 部门经理)
- ❑ 图书(书号, 书名, 出版日期, 出版社, 书架号, 房间号, 管理员)
- ❑ 客户(客户号, 客户名称, 类别, 联系电话, 欲购产品编码, 产品名称, 数量, 要货日期)

6.1 数据依赖理论

--函数依赖

➤ 主键和外来键

- $R(U)$ 中的任一候选键可以作为R的主键。
- 若 $R(U)$ 中的属性或属性组合X并非R的候选键，但X却是另一关系的候选键，则称X为R的外来键。

例如：合同(合同号，合同名，签订日期，**供应商名**)

供应商(供应商名，供应商地址，执照号，法人代表)

6.1 数据依赖理论

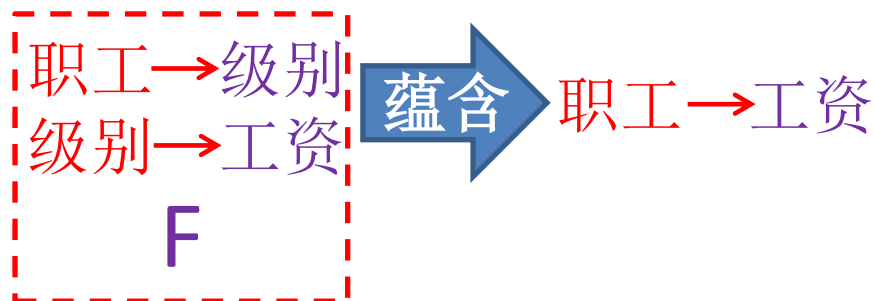
--函数依赖

➤ 逻辑蕴涵

设F是关系模式R(U)中的一个函数依赖集合，X, Y是R的属性子集，如果从F中的函数依赖能够推导出Y函数依赖于X，则称F逻辑蕴涵Y函数依赖于X，或称Y函数依赖于X是F的逻辑蕴涵。

职工表

职工	级别	工资
张一	4	800
张二	6	1500
张三	5	1000
张四	6	1500



6.1 数据依赖理论

--函数依赖

➤ 闭包

被F逻辑蕴涵的所有函数依赖集合是F的闭包, 记作 F^+ 。

例如: 设 $R=ABC$, $F=\{A\rightarrow B, B\rightarrow C\}$, 则 F^+ 的组成如下:

□ $F^+=\{\phi \rightarrow \phi, A \rightarrow \phi, A \rightarrow A, A \rightarrow B, A \rightarrow C,$
 $A \rightarrow AB, A \rightarrow BC, A \rightarrow ABC, \dots\},$

共有43个“ $X \rightarrow Y$ ”形式的依赖。

其中, ϕ 表示空属性集。



6.1 数据依赖理论

--函数依赖

➤ Armstrong公理

设 $R(U)$ 是属性集 $U=\{X, Y, Z\cdots\}$ 上的一个关系模式, F 为 $R(U)$ 的一组函数依赖, 记为 $R(U, F)$, 则有如下规则成立:

- 自反律: 若 $Y \subseteq X \subseteq U$, 则 $X \rightarrow Y$ 被 F 逻辑蕴涵。
- 增广律: 若 $X \rightarrow Y \in F$, 且 $Z \subseteq U$, 则 $XZ \rightarrow YZ$ 被 F 逻辑蕴涵。
- 传递律: 若 $X \rightarrow Y \in F$, 且 $Y \rightarrow Z$, 则 $X \rightarrow Z$ 被 F 逻辑蕴涵。

6.1 数据依赖理论

--函数依赖

➤ 由Armstrong公理可推出如下结论

- 合并律：若 $X \rightarrow Y$ 且 $X \rightarrow Z$ ，则 $X \rightarrow YZ$ 。
- 分解律：若 $X \rightarrow YZ$ ，则有 $X \rightarrow Y$ 以及 $X \rightarrow Z$ 。
- 伪传递律：若 $X \rightarrow Y$ 且 $WY \rightarrow Z$ ，则 $XW \rightarrow Z$ 。

6.1 数据依赖理论

--函数依赖

➤ 等价和覆盖

关系模式 $R\langle U, F \rangle$ 上的两个依赖集 F 和 G ，如果 $F^+ = G^+$ ，则称 F 和 G 是等价的，记做 $F \equiv G$ 。若 $F \equiv G$ ，则称 G 是 F 的一个覆盖，反之亦然。两个等价的函数依赖集在表达能力上是完全相同的。

➤ 最小函数依赖集

如果函数依赖集 F 满足下列条件，则称 F 为最小函数依赖集或最小覆盖。

- F 中的任何一个函数依赖的右部仅含有一个属性；
- F 中不存在这样一个函数依赖 $X \rightarrow A$ ，使得 F 与 $F - \{X \rightarrow A\}$ 等价；
- F 中不存在这样一个函数依赖 $X \rightarrow A$ ， X 有真子集 Z 使得 $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ 与 F 等价。

6.1 数据依赖理论

--函数依赖

➤ 最小函数依赖集算法

求最小函数依赖集分三步

□ 第一步:使F中的任何一个函数依赖的右部仅含有一个属性;

□ 第二步:去掉多余的函数依赖。

--从第一个函数依赖 $X \rightarrow Y$ 开始将其从F中去掉,然后在剩下的函数依赖中求X的闭包 X^+ ,看 X^+ 是否包含Y,若是,则去掉 $X \rightarrow Y$;否则不能去掉,依次做下去。直到找不到冗余的函数依赖;

□ 第三步:去掉各依赖左部多余的属性。

--一个一个地检查函数依赖左部非单个属性的依赖。

例如 $XY \rightarrow A$,若要判Y为多余的,则以 $X \rightarrow A$ 代替 $XY \rightarrow A$ 是否等价?若等价,则Y是多余属性,可以去掉。



6.1 数据依赖理论

--函数依赖

➤ 最小函数依赖集算法

举例：已知关系模式 $R\langle U, F\rangle$ ， $U=\{A, B, C, D, E, G\}$ ， $F=\{AB\rightarrow C, D\rightarrow EG, C\rightarrow A, BE\rightarrow C, BC\rightarrow D, CG\rightarrow BD, ACD\rightarrow B, CE\rightarrow AG\}$ ，求 F 的最小函数依赖集。

▣ 第一步：利用分解率，将所有的函数依赖变成右边都是单个属性的函数依赖，

得 F 为：

$F=\{AB\rightarrow C, D\rightarrow E, D\rightarrow G, C\rightarrow A, BE\rightarrow C, BC\rightarrow D, CG\rightarrow B, CG\rightarrow D, ACD\rightarrow B, CE\rightarrow A, CE\rightarrow G\}$ ；

6.1 数据依赖理论

--函数依赖

➤ 最小函数依赖集算法

▣ 第二步：去掉F中多余的函数依赖

A. 设 $AB \rightarrow C$ 为冗余的函数依赖

✓ 则去掉 $AB \rightarrow C$ ，得： $F_1 = \{D \rightarrow E, D \rightarrow G, C \rightarrow A, BE \rightarrow C, BC \rightarrow D, CG \rightarrow B, CG \rightarrow D, ACD \rightarrow B, CE \rightarrow A, CE \rightarrow G\}$

✓ 在 F_1 中求 $(AB)^+$ ：扫描 F_1 中各个函数依赖，找到左部为 AB 或 AB 子集的函数依赖，因为找不到这样的函数依赖。故终止。

✓ $(AB)^+ = AB$ 不包含 C ，故 $AB \rightarrow C$ 不是冗余的函数依赖，不能从 F_1 中去掉。

B. 设 $CG \rightarrow B$ 为冗余的函数依赖

... ..

6.1 数据依赖理论

--函数依赖

➤ 最小函数依赖集算法

- ▣ 第三步：去掉第二步得到的F中各函数依赖左边多余的属性（只检查左部不是单个属性的函数依赖）

由于 $C \rightarrow A$ ，函数依赖 $ACD \rightarrow B$ 中的属性A是多余的，去掉A得 $CD \rightarrow B$ 。

最小函数依赖集为：

$$F = \{AB \rightarrow C, D \rightarrow E, D \rightarrow G, C \rightarrow A, BE \rightarrow C, \\ BC \rightarrow D, CG \rightarrow D, CD \rightarrow B, CE \rightarrow G\}$$

6.1 数据依赖理论

--函数依赖

➤ 属性闭包

对 F ， F^+ 中所有 $X \rightarrow A$ 的 A 的集合称为 X 的闭包，记为 X^+ 。

➤ 用途：

- X^+ 表示所有 X 可以决定的属性。
- 若 X^+ 包含 R 的所有属性，则 X 是超键。
- 当 X 不可约时则为候选键。
- $X \rightarrow Y$ 属于 F^+ 的必要充分条件是： Y 是 X^+ 的子集。即不求 F^+ ，也可以判断某函数依赖是否属于 F^+ 。

6.1 数据依赖理论

--函数依赖

➤ 求解候选码基本算法的具体步骤(方法一)

□ 第1步：求关系模式 $R\langle U, F \rangle$ 的最小函数依赖集 F ;

□ 第2步：将属性分类为 U_L, U_R, U_B ;

(U_L 表示仅在函数依赖集中各依赖关系式左边出现的属性的集合;

U_R 表示仅在函数依赖集中各依赖关系式右边出现的属性的集合;

另记 $U_B = U - U_L - U_R$)

□ 第3步：计算候选码

✓若 $U_L \neq \Phi$, 则其中的所有属性都是主属性, 计算 U_L 的闭包

• 若 $U_L^+ = U$, 则 U_L 为 R 的唯一的候选码;

• 若 $U_L^+ \neq U$, 转第4步。

✓若 $U_L = \Phi$, 转第4步;

6.1 数据依赖理论

--函数依赖

➤ 求解候选码基本算法的具体步骤(方法一)

- ▣ 第4步：将UL依次与UB中的属性组合, 并分别计算闭包 UL^+
 - ✓若 $UL^+=U$, 则UL中的属性集就是一个候选码;
 - ✓若 $UL^+ \neq U$, 转第4步, 直至UB中的所有属性处理完。

简而言之：取最小依赖集，计算UL闭包，如果UL闭包包含全属性，则UL为唯一候选码，如果不包含，则依次与UB属性组合后再求闭包是否包含全属性。(UL为空时，直接取UB依次组合求闭包)

6.1 数据依赖理论

--函数依赖

➤ 求解候选码基本算法的具体步骤(方法一)

例：设有一关系模式 $U(A, B, C, D, E, H)$ ，其函数依赖集为
 $F = \{(A, D) \rightarrow B, (A, B) \rightarrow E, C \rightarrow D, B \rightarrow C, (A, C) \rightarrow H\}$ ，
计算 S 的候选码。

□ 第一步，计算最小函数依赖集

✓ 根据定义， F 已是最小函数依赖集。

□ 第二步，确定 UL , UR , UB

✓ 可得 $UL = \{A\}$, $UR = \{E, H\}$, $UB = \{B, C, D\}$

6.1 数据依赖理论

--函数依赖

➤ 求解候选码基本算法的具体步骤(方法一)

▣ 第三步，计算候选码

✓ $UL^+ = A^+ = A \neq U$ ；因此属性A不是U的候选码；

✓ 把UB中的属性逐个加入UL中, 分别计算闭包UL, 结果如下：

- $\{A, B\}^+ = \{A, B, E, C, D, H\} = U$
- $\{A, C\}^+ = \{A, C, D, B, E, H\} = U$
- $\{A, D\}^+ = \{A, D, B, C, E, H\} = U$

因此, (A, B)、(A, C)和(A, D)都是U的候选码, 其他属性或属性组不会再是候选码。

6.1 数据依赖理论

--函数依赖

➤ 求解候选码基本算法的具体步骤(方法二)

- 第1步：设定 $K=R$
 - 第2步：对于 K 中的每个属性 A , 计算属性集 $K-A$ 关于函数依赖集 F 的闭包 $\{K-A\}$
 - 第3步：
 - ✓如果 $\{K-A\}=R$, 则属性 A 不是主属性, 从 K 中消去;
 - ✓否则属性 A 是一个主属性, 保留在 K 中
- 最后 K 即为 R 的候选码。

算法概述: 有 N 个属性, 从1到 N 循环。 K 初始为全部属性, 每次循环时减去第 N 个属性, 如果 KF^+ 包含全部属性, 则 K 的值重新附值为 K 减去第 N 个属性后的值; 否则 K 仍为上次循环后的值。

6.1 数据依赖理论

--函数依赖

➤ 求解候选码基本算法的具体步骤(方法三)

对于给定的 $R(U)$ 和函数依赖集 F ,

可以将它的属性划分为4类:

- L类, 仅出现在 F 的函数依赖左部的属性。
- R类, 仅出现在 F 的函数依赖右部的属性。
- N类, 在 F 的函数依赖左部和右部均未出现的属性。
- LR类, 在 F 的函数依赖左部和右部两部均出现的属性。

6.1 数据依赖理论

--函数依赖

➤ 求解候选码基本算法的具体步骤(方法三)

根据以下定理和推论来求解候选码:

- ❑ 定理1:对于给定的关系模式R及其函数依赖集F, 若 $X(X \in R)$ 是L类属性, 则X必为R的任一候选码的成员。
- ❑ 推论1:对于给定的关系模式R及其函数依赖集F, 若 $X(X \in R)$ 是L类属性, 且 X^+ 包含了R的全部属性, 则X必为R的唯一候选码。
- ❑ 定理2:对于给定的关系模式R及其函数依赖集F, 若 $X(X \in R)$ 是R类属性, 则X不在任何候选码中。

6.1 数据依赖理论

--函数依赖

➤ 求解候选码基本算法的具体步骤(方法三)

根据以下定理和推论来求解候选码：

- ▣ 定理3: 设有关系模式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2: 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的所有属性, 则X是R的唯一候选码。

L、R、N、LR类, 根据定理, L、N类必为候选码之一, 如果 L^+ 包含全部R, 则L为唯一候选码。R类不在任何候选码中。
L+N类且 $(L+N)^+$ 包含所有R, 则(L+N)为唯一候选码。

第6章 数据库设计理论

理论：概念、性质/公理/定理/推论、证明、算法等

6.1 数据依赖理论

--关于属性之间关系的理论

6.2 关系范式理论

--关于关系模式设计的规范化形式的理论

6.3 模式分解理论(关系模式规范化方法)

--关于大模式在分解为小模式过程中的理论

6.4 小结

6.2 关系范式理论

--关系范式

- 范式是关系模式满足不同程度的规范化要求的标准
 - ▣ 满足最低程度要求的范式属于第一范式，简称1NF；
 - ▣ 在第一范式中进一步满足一些要求的关系属于第二范式，简称2NF，依次类推，还有3NF、BCNF、4NF、5NF等关系范式。
 - ▣ 对关系模式的属性间的函数依赖加以不同的限制就形成了不同的范式。
 - ▣ 范式是递进的，即如果是一个关系是1NF的，它比不是1NF的关系要好；同样，2NF的关系比1NF的关系要好等等，范式越高、规范化程度越高，关系模式就越好。

6.2 关系范式理论

--关系范式

➤ 1NF: 关系设计的第1范式原则

- 关系模式R的所有属性的域都是原子的(域的元素是不可分的单元), 则 $R \in 1NF$ 。
- 将非第一范式的关系转换为第一范式的关系非常简单, 只需要将所有数据项都分解成不可再分的最小数据项就可以了。

学号	姓名	住址	
		城市	街道
1101	张一	威海	文化西路
1102	张二	济南	世昌大道
1103	张三	青岛	哈工大路

不符合1NF

学号	姓名	城市	街道
1101	张一	威海	文化西路
1102	张二	济南	世昌大道
1103	张三	青岛	哈工大路

符合1NF

6.2 关系范式理论

--关系范式

➤ 1NF: 关系设计的第1范式原则

▣ 多值属性

S#	C#
S1	{C1, C2, C3}



S#	C#
S1	C1
S1	C2
S1	C3

▣ 复合属性

S#
CS001
CS002
CS003



S#	Dept
001	CS
003	CS
003	CS

不符合1NF

符合1NF

6.2 关系范式理论

--关系范式

➤ 2NF: 关系设计的第2范式原则

- ❑ 如果关系 $R \in 1NF$ ，并且 R 中每一个非主属性完全函数依赖于任一个候选码，则 $R \in 2NF$ 。
- ❑ 存在非主属性部分依赖(即, 不满足第2范式)的关系设计将会存在非受控冗余、插入异常和删除异常等问题。
- ❑ 不满足第2范式，则可以通过将该模式分解成两个或多个小的模式, 来使其满足第2范式。分解过程如下：
 - ✓ 用组成主码的属性集合的每一个子集作为主码构成一个表。
 - ✓ 对于每个表，将依赖于此主码的属性放置到此表中。

6.2 关系范式理论

--关系范式

➤ 2NF: 关系设计的第2范式原则

▣ 例如: $R(\text{学号}, \text{姓名}, \text{系名}, \text{系主任}, \text{课号}, \text{成绩})$

- ✓ 分析函数依赖关系, 很显然 {学号, 课号} 是候选键,
- ✓ 非主属性 “姓名” 仅函数依赖于 “学号”, 也就是 “姓名” 部分函数依赖于主码 (学号, 课号) 而不是完全依赖;
- ✓ 非主属性 “系名” 仅函数依赖于 “学号”, 也就是 “系名” 部分函数依赖于主码 (学号, 课号) 而不是完全依赖;
- ✓ 非主属性 “系主任” 仅函数依赖于 “学号”, 也就是 “系主任” 部分函数依赖于主码 (学号, 课号) 而不是完全依赖。

因此该模式不属于2NF

6.2 关系范式理论

--关系范式

➤ 2NF: 关系设计的第2范式原则

- ❑ 如果关系 $R \in 1NF$ ，并且 R 中每一个非主属性完全函数依赖于任一个候选码，则 $R \in 2NF$ 。
- ❑ 存在非主属性部分依赖(即, 不满足第2范式)的关系设计将会存在非受控冗余、插入异常和删除异常等问题。
- ❑ 不满足第2范式，则可以通过将该模式分解成两个或多个小的模式, 来使其满足第2范式。分解过程如下：
 - ✓ 用组成主码的属性集合的每一个子集作为主码构成一个表。
 - ✓ 对于每个表，将依赖于此主码的属性放置到此表中。

6.2 关系范式理论

--关系范式

➤ 2NF: 关系设计的第2范式原则

□ 例如: $R(\text{学号}, \text{姓名}, \text{系名}, \text{系主任}, \text{课号}, \text{成绩})$

✓ 我们可以将R分解为两个关系模式

$R_1(\text{学号}, \text{课号}, \text{成绩})$, 主码为(学号, 课号)

$R_2(\text{学号}, \text{姓名}, \text{系名}, \text{系主任})$, 主码为(学号)

则 $R_1 \in 2NF$, $R_2 \in 2NF$ 。

说明: 第二范式消除了非主属性对候选键的部分依赖

6.2 关系范式理论

--关系范式

➤ 2NF: 关系设计的第2范式原则

- ❑ 选课关系表R(学号, 姓名, 年龄, 课程名称, 成绩, 学分), 关键字为组合关键字(学号, 课程名称), 不符合2NF, 这个选课关系表会存在如下问题:

✓数据冗余:

同一门课程由n个学生选修, “学分”就重复n-1次;

同一个学生选修了m门课程, 姓名和年龄就重复了m-1次。

✓更新异常:

若调整了某门课程的学分, 数据表中所有行的“学分”值都要更新, 否则会出现同一门课程学分不同的情况。

6.2 关系范式理论

--关系范式

➤ 2NF: 关系设计的第2范式原则

✓插入异常:

假设要开设一门新的课程，暂时还没有人选修。这样，由于还没有“学号”关键字，课程名称和学分也无法记录入数据库。

✓删除异常:

假设一批学生已经完成课程的选修，这些选修记录就应该从数据库表中删除。但是，与此同时，课程名称和学分信息也被删除了。很显然，这也会导致插入异常。

6.2 关系范式理论

--关系范式

➤ 3NF: 关系设计的第3范式原则

- 如果关系 $R \in 2NF$ ，并且 R 中每一个非主属性对任何候选码都不存在传递函数依赖，则 $R \in 3NF$ 。
- 简而言之，第三范式就是属性不依赖于其它非主属性。因此，满足第三范式的数据库表应该不存在如下依赖关系：
关键字段 \rightarrow 非关键字段 $x \rightarrow$ 非关键字段 y
- 存在非主属性传递依赖于候选键（即，不满足第3NF）的关系设计将会存在非受控冗余、插入异常和删除异常等问题。
- 不满足第3范式，则可以通过将该模式分解成两个或多个小模式，来使其满足第3范式。

6.2 关系范式理论

--关系范式

➤ 3NF: 关系设计的第3范式原则

▣ 去掉传递函数依赖的分解过程如下:

- ✓ 对于不是候选码的每个决定因子，从关系模式中删除依赖于该决定因子的属性。
- ✓ 新建一个关系模式，新的关系模式中应包含在原表中所有依赖于该决定因子的属性。
- ✓ 将决定因子作为新关系模式的主码。

6.2 关系范式理论

--关系范式

➤ 3NF: 关系设计的第3范式原则

▣ 例如: $R(\text{学号}, \text{姓名}, \text{系名}, \text{系主任}, \text{课号}, \text{成绩})$

✓ R 分解为两个关系模式

$R_1(\text{学号}, \text{课号}, \text{成绩})$, 主码为 $(\text{学号}, \text{课号})$

$R_2(\text{学号}, \text{姓名}, \text{系名}, \text{系主任})$, 主码为 (学号)

则 $R_1 \in 2NF$, $R_2 \in 2NF$ 。

✓ 但在 $R_2(\text{学号}, \text{姓名}, \text{系名}, \text{系主任})$ 中, 存在如下函数依赖:

$\text{学号} \rightarrow \text{系名}$ $\text{系名} \rightarrow \text{系主任}$ $\text{系名} \nrightarrow \text{学号}$

那么, 存在着一个传递函数依赖 “ $\text{学号} \rightarrow \rightarrow \text{系主任}$ ” 成立。

因此该模式不属于 3NF

6.2 关系范式理论

--关系范式

➤ 3NF: 关系设计的第3范式原则

▣ 去掉传递函数依赖的分解过程如下分解过程如下:

- ✓ 对于不是候选码的每个决定因子，从关系模式中删除依赖于该决定因子的属性。
- ✓ 新建一个关系模式，新的关系模式中应包含在原表中所有依赖于该决定因子的属性。
- ✓ 将决定因子作为新关系模式的主码。

6.2 关系范式理论

--关系范式

➤ 3NF: 关系设计的第3范式原则

▣ 例如: R(学号, 姓名, 系名, 系主任, 课号, 成绩)

✓ 将R2分解为

R21(学号, 姓名, 系名)

R22(系名, 系主任)

则这两个关系模式不再存在传递依赖, 它们均为第三范式。

6.2 关系范式理论

--关系范式

➤ 3NF: 关系设计的第3范式原则

- ❑ 再例如：存在一个部门信息表，其中每个部门有部门编号、部门名称、部门简介等信息。
- ❑ 那么在的员工信息表中列出部门编号后就不能再将部门名称、部门简介等与部门有关的信息再加入员工信息表中。
- ❑ 如果不存在部门信息表，则根据第三范式(3NF)也应该构建它，否则就会有大量的数据冗余(自行分析)。

因此在通常的数据库设计中，一般要求要达到3NF。
3NF是一个实际可用的关系模式应满足的最低范式。

6.2 关系范式理论

--关系范式

➤ BCNF: 关系设计的Boyce-Codd范式原则

- ❑ 如果关系模式 $R(U, F)$ 的所有属性(包括主属性和非主属性)都不传递依赖于 R 的任何候选关键字, 那么称关系 R 是属于BCNF的。
- ❑ BCNF范式比3NF又进了一步, 通常认为是修正的第三范式; 关系设计满足Boyce-Codd范式, 则其一定满足第3范式; 反之则不然。
3NF——只消除非主属性对主属性的传递依赖;
BCNF——消除所有属性对主属性的传递依赖。

6.2 关系范式理论

--关系范式

➤ BCNF: 关系设计的Boyce-Codd范式原则

□ 例如: $R(\text{学生}, \text{教师}, \text{课程})$,

假设: 每一个教师只教一门课, 每门课有若干个教师,
某一学生选定某门课, 就对应一个固定的教师。

由语义可得到如下函数依赖:

✓ $(\text{学生}, \text{课程}) \rightarrow \text{教师}$; $(\text{学生}, \text{教师}) \rightarrow \text{课程}$; $\text{教师} \rightarrow \text{课程}$
 $(\text{学生}, \text{课程}), (\text{学生}, \text{教师})$ 都是候选码。

R 是 3NF, 因为没有任何非主属性对码传递依赖或部分依赖。

但 R 不是 BCNF 关系, 因为课程传递依赖于 $(\text{学生}, \text{课程})$, 而 $(\text{学生}, \text{课程})$ 是候选关键字。

6.2 关系范式理论

--关系范式

➤ BCNF: 关系设计的Boyce-Codd范式原则

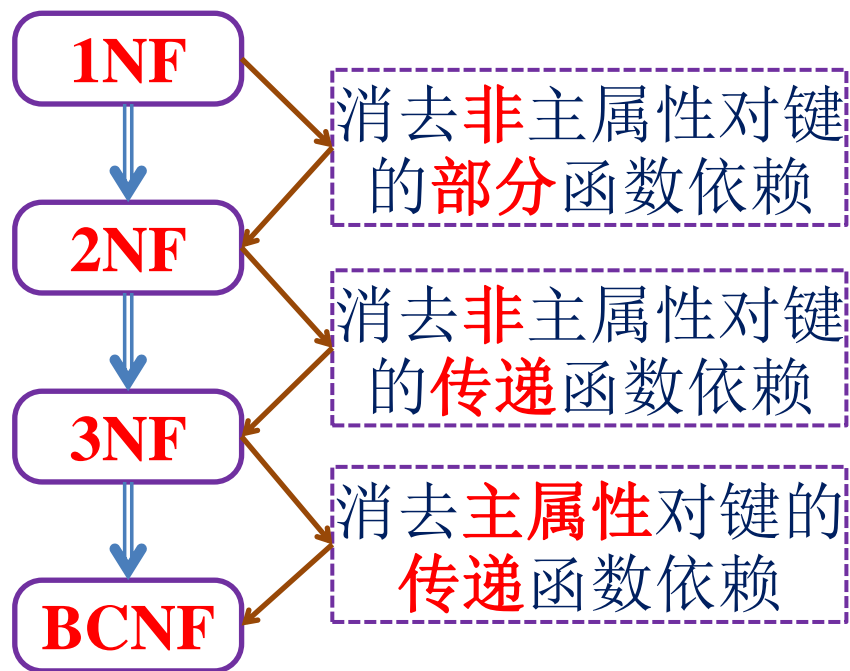
- ❑ 存在非主属性对非候选键的函数依赖的不良特性
 - ✓ 插入异常: 如果没有学生选修某位老师的任课, 则该老师担任课程的信息就无法插入
 - ✓ 删除异常: 删除学生选课信息, 会删除掉老师的任课信息
 - ✓ 数据冗余: 每位学生都存储了有关老师所教授的课程的信息
 - ✓ 更新异常: 如果老师所教授的课程有所改动, 则所有选修该老师课程的学生元组都要做改动

6.2 关系范式理论

--关系范式

➤ 四种范式之间存在如下关系

$$\text{BCNF} \subseteq 3\text{NF} \subseteq 2\text{NF} \subseteq 1\text{NF}$$



第6章 数据库设计理论

理论：概念、性质/公理/定理/推论、证明、算法等

6.1 数据依赖理论

--关于属性之间关系的理论

6.2 关系范式理论

--关于关系模式设计的规范化形式的理论

6.3 模式分解理论(关系模式规范化方法)

--关于大模式在分解为小模式过程中的理论

6.4 小结

6.3 模式分解理论

--模式分解

➤ 模式分解的定义

▣ 关系模式 $R\langle U, F \rangle$ 的一个分解是指用 R 的一组子集

$\rho = \{R_1\langle U_1, F_1 \rangle, R_2\langle U_2, F_2 \rangle, \dots \dots, R_n\langle U_n, F_n \rangle\}$ 来代替它。

其中 $U = U_1 \cup U_2 \cup \dots \cup U_k$; $U_i \not\subseteq U_j (i \neq j)$, F_i 是 F 在 U_i 上的投影。

注：后面我们用 R_i 代替 $R_i(U_i, F_i)$, R 代替 $R(U, F)$ 。

▣ 对于关系模式 R 的任一关系 r , 它向 ρ 的投影连接记为 $m_\rho(r)$

$$m_\rho(r) = \pi_{R_1}(r) \bowtie \dots \bowtie \pi_{R_k}(r) = \bigotimes_{(i=1, 2, \dots, k)} \pi_{R_i}(r)$$

这里: $\pi_{R_i}(r) = \{t[R_i] \mid t \in r, i=1, \dots, k\}$

6.3 模式分解理论

--模式分解

➤ 模式分解的方法

▣ 分解成2NF模式集的算法

设关系模式 $R(U)$ ，主码是 W ， R 上还存在 $X \rightarrow Z$ ，并且 Z 是非主属性和 $X \subset W$ ，那么 $W \rightarrow Z$ 就是非主属性对码的部分依赖。此时，应把 R 分解成两个关系模式：

$R_1(XZ)$ ，主码是 X ；

$R_2(Y)$ ，其中 $Y=U-Z$ ，主码仍为 W ，外码是 X （参照 R_1 ）

利用外码和主码的连接可以从 R_1 和 R_2 重新得到 R 。

如果 R_1 和 R_2 还不是2NF，则重复上述过程，一直到数据库模式中的每个关系模式都是2NF为止。

6.3 模式分解理论

--模式分解

➤ 模式分解的方法

▣ 分解成3NF模式集的算法

设关系模式 $R(U)$, 主码是 W , R 上还存在 $X \rightarrow Z$, 并且 Z 是非主属性, $Z \not\subseteq X$, X 不是候选码, 那么 $W \rightarrow Z$ 就是非主属性对码的传递依赖。此时, 应把 R 分解成两个关系模式:

$R_1(XZ)$, 主码是 X ;

$R_2(Y)$, 其中 $Y=U-Z$, 主码仍为 W , 外码是 X (参照 R_1)

利用外码和主码的连接可以从 R_1 和 R_2 重新得到 R 。

如果 R_1 和 R_2 还不是3NF, 则重复上述过程, 一直到数据库模式中的每个关系模式都是3NF为止。

6.3 模式分解理论

--模式分解

➤ 模式分解的方法

▣ 无损且保持函数依赖分解成3NF模式集的合成算法

1. 对于关系模式R和R上成立的函数依赖集F, 先求出F的最小依赖集, 然后再把最小依赖集中那些左部相同的函数依赖用合并性合并起来。
2. 对最小依赖集中, 每个函数依赖 $X \rightarrow Y$ 去构成一个模式XY。
3. 在构成的模式集中, 如果每个模式都不包含R的侯选码, 那么把侯选码作为一个模式放入模式集中。

注: 这样得到的模式集是关系模式R的一个分解,
并且这个分解即是无损分解, 又保持了函数依赖。

6.3 模式分解理论

--模式分解

➤ 模式分解的方法

▣ 无损分解成BCNF模式集的算法

1. 令 $\rho = \{R\}$ 。
2. 对每个模式 $s \in \rho$, 若 $s \notin \text{BCNF}$, 则 s 上必有 $X \rightarrow A$ 成立且 X 不是 s 的超键且 $A \notin X$, 此时用模式 s_1, s_2 替代 ρ 中的模式 s , 其中 s_1 由 A 和 X 构成, s_2 由 s 中除 A 以外的所有属性构成(可以发现, $s_1 \in \text{BCNF}$)。
3. 重复步骤2, 直至 ρ 中全部关系模式达到BCNF。

注：本算法不能保证一关系模式分解成BCNF而又保持依赖。

6.3 模式分解理论

--模式分解

➤ 模式分解需要关注:

- ▣ R与 ρ 在数据内容方面是否等价: 分解的无损连接性;
- ▣ R与 ρ 在数据依赖方面是否等价: 分解的保持依赖性。

➤ 无损分解的定义

- ▣ $\rho = \{R_1\langle U_1, F_1 \rangle, R_2\langle U_2, F_2 \rangle, \dots, R_n\langle U_n, F_n \rangle\}$ 是关系模式 $R\langle U, F \rangle$ 的一个分解, 如果对于R的任何满足函数依赖集F的关系r, 有 $r = m_\rho(r)$, 则称 ρ 是R相对于F的一个无损连接分解。

$$m_\rho(r) = \pi_{R_1}(r) \bowtie \dots \bowtie \pi_{R_k}(r) = \bigotimes_{(i=1, 2, \dots, k)} \pi_{R_i}(r)$$

6.3 模式分解理论

--模式分解

➤ 保持依赖分解的定义

- $\rho = \{R_1\langle U_1, F_1 \rangle, R_2\langle U_2, F_2 \rangle, \dots, R_n\langle U_n, F_n \rangle\}$ 是关系模式 $R\langle U, F \rangle$ 的一个分解, 如果在 $R_i(F)$ 中的所有依赖之并集 ($i=1, \dots, k$) 逻辑蕴涵 F 的每个依赖, 则称分解 ρ 保持依赖集 F 。
- 其中 $\pi_{R_i}(F)$ 是 F 在 R_i 上的投影, 即 F 中的任一投影 $X \rightarrow Y$, 如果 X, Y 均包含于 R_i , 则 $X \rightarrow Y \in R_i(F)$ 。 R_i 指 R 的属性集。

6.3 模式分解理论

--模式分解

➤ 保持依赖分解与保持无损分解的关系

一个无损连接的分解不一定具有函数依赖保持性，
同样，一个保持函数依赖的分解也不一定具有无损连接性，
而且有些实用的分解算法也往往顾此失彼，难以兼顾。

例如： $R(CSZ)$, $F = \{CS \rightarrow Z, Z \rightarrow C\}$, C 是城市, S 是街区, Z 是邮政编码, $\rho = \{R_1(SZ), R_2(CZ)\}$ 为一无损连接分解, 但却不保持依赖。

再例如： $R(ABCD)$, $F = \{A \rightarrow B, C \rightarrow D\}$, $\rho = \{R_1(AB), R_2(CD)\}$ 为一保持依赖分解, 但不是无损连接分解。

6.3 模式分解理论

--模式分解

➤ 无损连接性检验算法:

□Input: 关系模式 $R=A_1A_2\cdots A_n$, 函数依赖集 F ,
分解 $\rho=\{R_1, \cdots, R_k\}$

□Output: ρ 是否是无损连接的判断

□Method:

1. 构造一 k 行 n 列的表, 可称为 R_ρ 表。
其中第 j 列对应于 A_j , 第 i 行对应于 R_i ,
若 $A_j \in R_i$, 则 R_ρ 表中第 i 行第 j 列位置
填写符号 a_j , 否则填写 b_{ij} 。

	$A_1 \dots A_j \dots A_n$
R_1	
\dots	
R_i	
\dots	
R_k	

[返回](#)

6.3 模式分解理论

--模式分解

➤ 无损连接性检验算法:

2. 根据 $\forall (X \rightarrow Y) \in F$, 对 $R\rho$ 表进行修改:

- ✓ 考察 F 中的每一个函数依赖 $X \rightarrow Y$, 在属性 X 所在的那些列上寻找具有相同符号的行;
- ✓ 如果找到这样的两行或更多的行, 则使这些行上属性 Y 所在的列上元素相同;
- ✓ 如果其中有一个为 a_j , 则这些元素均变为 a_j ; 否则改为 b_{mj} , 其中 m 为这些行的最小行号。
- ✓ 注意, 若某个 b_{ij} 被改动, 则该列中凡与 b_{ij} 相同的符号均作相同的改动。
- ✓ 可以循环地对 F 中的函数依赖进行逐个处理, 直到表的内容不能再被修改为止。

[返回](#)

6.3 模式分解理论

--模式分解

➤ 无损连接性检验算法:

3. 在上述修改的表中, 如果发现有一行变成 a_1, a_2, \dots, a_n (全 a), 则 ρ 是无损连接分解, 否则 ρ 是有损连接分解。

[返回](#)

6.3 模式分解理论

--模式分解

➤ 无损连接性检验算法：

例：已知关系模式 $R(ABCDE)$ 及其函数依赖集

$$F = \{A \rightarrow C, B \rightarrow C, C \rightarrow D, DE \rightarrow C, CE \rightarrow A\},$$

试检验分解

$$\rho = \{R_1(AD), R_2(AB), R_3(BE), R_4(CDE), R_5(AE)\}$$

是否为无损连接。

6.3 模式分解理论

--模式分解

➤ 无损连接性检验算法:

解: (1) 构造 R_ρ 表

	A	B	C	D	E
R1					
R2					
R3					
R4					
R5					

关系模式 $R(ABCDE)$

函数依赖集 $F=\{A \rightarrow C, B \rightarrow C, C \rightarrow D, DE \rightarrow C, CE \rightarrow A\}$,

分解 $\rho=\{R1(AD), R2(AB), R3(BE), R4(CDE), R5(AE)\}$

(2) 用 $A \rightarrow C$ 修改 R_ρ 表: A列上R1R2R5行上的值相同, 设置C列上R1R2R5行上的值相同, 因为没有 a_j 存在, 所以选任一个 b_{ij} 进行设置均可, 假设设置为 b_{13} 。

	A	B	C	D	E
R1	a1	b12	b13	a4	b15
R2	a1	a2	b13	b24	b25
R3	b31	a2	b33	b34	a5
R4	b41	b42	a3	a4	a5
R5	a1	b52	b13	b54	a5

6.3 模式分解理论

--模式分解

➤ 无损连接性检验算法:

解: (1) 构造 R_ρ 表

	A	B	C	D	E
R1	a1	b12	b13	a4	b15
R2	a1	a2	b23	b24	b25
R3	b31	a2	b33	b34	a5
R4	b41	b42	a3	a4	a5
R5	a1	b52	b53	b54	a5

关系模式 $R(ABCDE)$

函数依赖集 $F=\{A \rightarrow C, B \rightarrow C, C \rightarrow D, DE \rightarrow C, CE \rightarrow A\}$,

分解 $\rho=\{R1(AD), R2(AB), R3(BE), R4(CDE), R5(AE)\}$

(2) 用 $A \rightarrow C$ 修改 R_ρ 表: A列上R1R2R5行上的值相同, 设置C列上R1R2R5行上的值相同, 因为没有 a_j 存在, 所以选任一个 b_{ij} 进行设置均可, 假设设置为 b_{13} 。

	A	B	C	D	E
R1	a1	b12	b13	a4	b15
R2	a1	a2	b13	b24	b25
R3	b31	a2	b33	b34	a5
R4	b41	b42	a3	a4	a5
R5	a1	b52	b13	b54	a5

6.3 模式分解理论

--模式分解

➤ 无损连接性检验算法:

(3) 用 $B \rightarrow C$ 继续修改 $R\rho$ 表: B列上R2R3行上的值相同, 设置C列上R2R3行上的值相同, 设置为 b_{13} 。
(尽可能保持以前的相等关系)

	A	B	C	D	E
R1	a1	b12	b13	a4	b15
R2	a1	a2	b13	b24	b25
R3	b31	a2	b13	b34	a5
R4	b41	b42	a3	a4	a5
R5	a1	b52	b13	b54	a5

(4) 用 $CE \rightarrow A$ 继续修改 $R\rho$ 表: CE两列上R3R5行上的值全相同, 设置A列上R3R5行上的值相同, 设置为a1。

	A	B	C	D	E
R1	a1	b12	b13	a4	b15
R2	a1	a2	b13	b24	b25
R3	a1	a2	b13	b34	a5
R4	b41	b42	a3	a4	a5
R5	a1	b52	b13	b54	a5

6.3 模式分解理论

--模式分解

➤ 无损连接性检验算法:

(5) 用 $C \rightarrow D$, $DE \rightarrow C$ 继续修改 R_ρ 表, 并循环操作, 直到 R_ρ 表不再改变为止。

	A	B	C	D	E
R1	a1	b12	a3	a4	b15
R2	a1	a2	a3	a4	b25
R3	a1	a2	a3	a4	a5
R4	b41	b42	a3	a4	a5
R5	a1	b52	a3	a4	a5

(6) 在结果 R_ρ 表可以发现有一行 R_3 全为 a , 则说明 ρ 是无损连接。

	A	B	C	D	E
R1	a1	b12	a3	a4	b15
R2	a1	a2	a3	a4	b25
R3	a1	a2	a3	a4	a5
R4	b41	b42	a3	a4	a5
R5	a1	b52	a3	a4	a5

6.3 模式分解理论

--模式分解

➤ 保持依赖性检验算法:

▣ **Input:** 关系模式 $R=A_1A_2\cdots A_n$, R 上的函数依赖集 F ,
分解 $\rho=\{R_1, \cdots, R_k\}$

▣ **Output:** ρ 否是保持依赖的判断

▣ **Method:** 令 $G= \bigcup_{(i=1,\dots,k)} \pi_{R_i}(F)$, 只需检查 G 是否覆盖 F 即可。

1. 首先对每个 $X \rightarrow Y \in F$ 计算 G 中的 X_G^+ ,
(如果 X 不包含于 R_i 则不需计算了)

$Z=X$ WHILE Z 的变化发生 DO

FOR $i=1$ to k DO $Z=Z \cup ((Z \cap R_i)^+ \cap R_i)$

6.3 模式分解理论

--模式分解

➤ 保持依赖性检验算法:

1. 首先对每个 $X \rightarrow Y \in F$ 计算 G 中的 X_G^+ ,
(如果 X 不包含于 R_i 则不需计算了)
Z=X WHILE Z的变化发生 DO
FOR i=1 to k DO $Z = Z \cup ((Z \cap R_i)^+ \cap R_i)$
2. 判断 G 是否逻辑蕴涵 $X \rightarrow Y$: 前面计算的结果 Z 便是 X^+ , 如果 Z 包含 Y , 则 G 逻辑蕴涵 $X \rightarrow Y$, 否则便不逻辑蕴涵。
3. 判断 ρ 是否保持依赖: 如果 G 逻辑蕴涵 F 中的每一个函数依赖, 则说 ρ 是保持依赖的分解, 否则便不是保持依赖的分解。

6.3 模式分解理论

--模式分解

➤ 保持依赖性检验算法：

例：设关系模式 $R = \{CITY, ST, ZIP\}$

表示各城市街道的邮政编码，

其中属性分别表示城市、街道名和邮政编，

$$F = \{ (CITY, ST) \rightarrow ZIP, ZIP \rightarrow CITY \}。$$

如果将 R 分解为

$$\rho = \{R_1, R_2\}, \text{ 其中 } R_1 = \{ST, ZIP\}, R_2 = \{CITY, ZIP\}$$

判断 ρ 的依赖保持性。

6.3 模式分解理论

--模式分解

➤ 保持依赖性检验算法：

解： $F1 = \pi_{R1}(F) = \{\Phi\}$

$F2 = \pi_{R2}(F) = \{ZIP \rightarrow CITY\}$

$F1 \cup F2 = \{ZIP \rightarrow CITY\}$

容易验证 $F1 \cup F2$ 不蕴涵 F 中的 $(CITY, ST) \rightarrow ZIP$,

所以 ρ 不是保持函数依赖的分解。

但是按照无损连接性检验算法推算, ρ 是无损连接分解。

关系模式规范化实战

一、实例：

假设某商业集团数据库中有一关系模式R（商店编号，商品编号，数量，部门编号，负责人），如果规定：

1. 每个商店的每种商品只在一个部门销售；
2. 每个商店的每个部门只有一个负责人；
3. 每个商店的每种商品只有一个库存数量。

试回答下列问题：

1. 根据上述规定，写出关系模式R的基本函数依赖；
2. 找出关系模式R的候选关键字；
3. 试问关系模式R最高已经达到第几范式？为什么？
4. 如果R已达3NF，是否已达BCNF？若不是BCNF，将其分解为BCNF模式集。

二、预处理：

为了方便，我们用代号代表每个属性：

A—商店编号

B—商品编号

C—部门编号

D—数量

E—负责人

这样，有关系模式：

$R(U, F)$

$U = \{A, B, C, D, E\}$

三、根据上述规定，写出关系模式R的基本函数依赖：

为了消除关系模式在操作上的异常问题，优化数据模式，我们需要对关系模式进行规范化处理。

首先需要做的就是函数依赖，以便能确切地反映实体内部各属性间的联系。经过对数据语义的分析我们得出下面的依赖关系：

1. 语义：每个商店的每种商品只在一个部门销售，即已知商店和商品名称可以决定销售部门

例：东店——海尔洗衣机——一定在家电部销售

所以得出函数依赖： $AB \rightarrow C$

2. 语义：每个商店的每个部门只有一个负责人，即已知商店和部门名称可以决定负责人

例：东店——家电部——部门经理一定是张三

所以得出函数依赖： $AC \rightarrow E$

三、根据上述规定，写出关系模式R的基本函数依赖：

为了消除关系模式在操作上的异常问题，优化数据模式，我们需要对关系模式进行规范化处理。

而首先需要做的就是函数依赖，以便能确切地反映实体内部各属性间的联系。经过对数据语义的分析我们得出下面的依赖关系：

3. 语义：每个商店的每种商品只有一个库存数量，即已知商店和商品名称可以决定库存数量

例：东店——海尔洗衣机——库存10台

所以得出函数依赖是： $AB \rightarrow D$

这样：

在关系模式R (U, F) 中，

基本函数依赖集是： $F = \{ AB \rightarrow C, AC \rightarrow E, AB \rightarrow D \}$ 。

四、找出关系模式R的候选关键字：

1. 根据分解性先分解所有依赖的右边为单属性：

可以看出： $F = \{ AB \rightarrow C, AC \rightarrow E, AB \rightarrow D \}$ 中所有依赖的右边已为单属性。

2. 对所有依赖的左边为多属性的情况，消除左侧冗余属性：

下面计算判断 $AB \rightarrow C$ 中**有无无关属性**：

① 设 $A \rightarrow C$ ，在 $F = \{ AB \rightarrow C, AC \rightarrow E, AB \rightarrow D \}$ 中计算A的闭包 A^+ ：

首先，初始化 $A^+ = \{A\}$ ；

经观察，在 $F = \{ AB \rightarrow C, AC \rightarrow E, AB \rightarrow D \}$ 中，属性A不能“带进”任何属性。即A的闭包 A^+ 就是 $\{A\}$ ，也就是 $A^+ = \{A\}$ ，
所以 $AB \rightarrow C$ 中B不是无关属性。

四、找出关系模式R的候选关键字：

1. 根据分解性先分解所有依赖的右边为单属性：

可以看出： $F = \{ AB \rightarrow C, AC \rightarrow E, AB \rightarrow D \}$ 中所有依赖的右边已为单属性。

2. 对所有依赖的左边为多属性的情况，消除左侧冗余属性：

下面计算判断 $AB \rightarrow C$ 中**有无无关属性**：

② 设 $B \rightarrow C$ ，在 $F = \{ AB \rightarrow C, AC \rightarrow E, AB \rightarrow D \}$ 中计算B的闭包 B^+ ：

首先，初始化 $B^+ = \{B\}$ ；

经观察，在 $F = \{ AB \rightarrow C, AC \rightarrow E, AB \rightarrow D \}$ 中，属性B不能“带进”任何属性。即B的闭包 B^+ 就是 $\{B\}$ ，也就是 $B^+ = \{B\}$ ，

所以 $AB \rightarrow C$ 中A不是无关属性。

同理， $AC \rightarrow E$ 和 $AB \rightarrow D$ 中左边亦无无关属性。

四、找出关系模式R的候选关键字：

3. 下面计算在 $F = \{AB \rightarrow C, AC \rightarrow E, AB \rightarrow D\}$ 中有无冗余依赖：

我们去掉 $AB \rightarrow C$ ，依赖集变为 $F = \{AC \rightarrow E, AB \rightarrow D\}$ 。

① 首先，初始化 $\{AB\}^+ = \{A, B\}$ ；

② 在 $F = \{AC \rightarrow E, AB \rightarrow D\}$ 中，有 $AB \rightarrow D$ ，即AB可以“带进”D属性，这时 $\{AB\}^+ = \{A, B, D\}$ ；

③ 经观察已不能再“带进”其它属性。

即 $\{AB\}$ 的闭包 $\{AB\}^+$ 就是 $\{A, B, D\}$ ，也就是 $\{AB\}^+ = \{A, B, D\}$ 。

因为 $\{A, B, D\}$ 中不包含C，所以我们说 $AB \rightarrow C$ 不是冗余依赖。

同理计算， $AC \rightarrow E$ 和 $AB \rightarrow D$ 亦不是冗余依赖。

到此，确定 $F = \{AB \rightarrow C, AC \rightarrow E, AB \rightarrow D\}$ 已是最小函数依赖集。

四、找出关系模式R的候选关键字：

4. 寻找候选关键字：

在 $F = \{AB \rightarrow C, AC \rightarrow E, AB \rightarrow D\}$ 中，我们对所有属性进行归类如下：

L类属性，即仅在依赖左边出现的属性：A, B

R类属性，即仅在依赖右边出现的属性：E, D

LR类属性，即既在依赖左边又在依赖右边出现的属性：C

N类属性，即既不在依赖左边又不在依赖右边出现的属性：无

我们知道，L类属性和N类属性一定在候选关键字中，

R类属性一定不在候选关键字中。

所以，A, B一定在候选关键字中，

E, D一定不在候选关键字中。

这是定性的结果



四、找出关系模式R的候选关键字：

4. 寻找候选关键字：

具体的候选关键字是什么呢？

- ① 首先，计算L类属性AB的闭包： $\{A, B\}^+ = \{A, B, D, C, E\}$ ，
- ② 因为AB的闭包 $\{A, B, D, C, E\}$ 已经包含了所有R的属性，
- ③ 所以， $\{A, B\}$ 是唯一候选关键字。

对于LR类属性参与候选关键字的相关计算稍嫌复杂，但这里已经找出了关系模式R $\{A, B, C, D, E\}$ 的唯一候选关键字。



五、关系模式R最高已经达到第几范式？为什么？

很明显，关系模式R(A, B, C, D, E)中的所有属性值都是不可再分的原子项，所以该关系模式已满足第一范式。

那么关系模式R(A, B, C, D, E)是否满足2NF？

根据范式的相关定义我们得知：如果关系模式R(U, F)中的所有非主属性都完全函数依赖于任一候选关键字，则该关系是第二范式。

从上面的分析我们知道R(A, B, C, D, E)的唯一候选关键字是{A, B}；非主属性是：C、D、E；函数依赖集是{ $AB \rightarrow C$, $AC \rightarrow E$, $AB \rightarrow D$ }。

所以：

$AB \rightarrow C$

例：东店——海尔洗衣机——一定在家电部销售

五、关系模式R最高已经达到第几范式？为什么？

所以：

$AB \rightarrow D$

例：东店——海尔洗衣机(只在家电部销售)——库存10台

$AB \rightarrow E (AB \rightarrow C, AC \rightarrow E)$

例：东店——海尔洗衣机(——卖海尔洗衣机的部门——家电部)——部门经理是张三

关系模式R(A, B, C, D, E)满足2NF。

进一步分析：

非主属性C、D、E之间不存在相互依赖，即关系模式R(A, B, C, D, E)不存在非主属性对候选关键字的传递依赖，根据第三范式的定义，关系模式R(A, B, C, D, E)已满足3NF。

六、R已达3NF，是否已达BCNF？若不是BCNF，将其分解为BCNF模式集。

由BCNF的定义得知：如果关系模式每个决定因素都包含关键字(而不是被关键字所包含)，则R满足BCNF。

分析：

在 $F = \{AB \rightarrow C, AC \rightarrow E, AB \rightarrow D\}$ 中，

有依赖 $AC \rightarrow E$ 的左边 $\{A, C\}$ 不包含候选关键字 $\{A, B\}$ ，

即 $AC \rightarrow E$ 是不满足BCNF的要求。

所以，关系模式 $R(A, B, C, D, E)$ 不满足BCNF。



六、R已达3NF，是否已达BCNF？若不是BCNF，将其分解为BCNF模式集。

1. 分解关系模式 $R(A, B, C, D, E)$, $F = \{AB \rightarrow C, AC \rightarrow E, AB \rightarrow D\}$ ：

分解3NF, 从不满足BCNF条件的依赖 $AC \rightarrow E$ 入手，我们得到两个新关系模式： $R_1(A, C, E)$ 和 $R_2(A, C, B, D)$

R_1 由不满足BCNF条件依赖 $AC \rightarrow E$ 的所有属性组成，

R_2 由依赖 $AC \rightarrow E$ 的决定因素和R的其余属性组成。

根据6.3无损分解成BCNF范式集的算法章节所讲解的内容可知 R_1 满足BCNF的要求，但是 R_2 满足不足呢？



六、R已达3NF，是否已达BCNF？若不是BCNF，将其分解为BCNF模式集。

2. 证明R1满足BCNF：

对于一个新关系，不知道它的依赖集，不知道它的候选关键字，我们需要借助原R(A, B, C, D, E)的依赖集 $F = \{AB \rightarrow C, AC \rightarrow E, AB \rightarrow D\}$ 。

① 对R1(A, C, E)：

首先，写出R1的所有子集

是： $\{A\}, \{C\}, \{E\}, \{A, C\}, \{A, E\}, \{C, E\}$

在 $F = \{AB \rightarrow C, AC \rightarrow E, AB \rightarrow D\}$ 中：

子集 $\{A\}$ 的闭包 $\{A\}^+ = \{A\}$ ，闭包中无不属于子集 $\{A\}$ 的“新属性”，所以找不到新依赖。



六、R已达3NF，是否已达BCNF？若不是BCNF，将其分解为BCNF模式集。

2. 证明R1满足BCNF：

对于一个新关系，不知道它的依赖集，不知道它的候选关键字，我们需要借助原R(A, B, C, D, E)的依赖集 $F = \{AB \rightarrow C, AC \rightarrow E, AB \rightarrow D\}$ 。

① 对R1(A, C, E)：

子集{C}的闭包 $\{C\}^+ = \{C\}$ ，闭包中无不属于子集{C}的“新属性”，所以找不到新依赖。

子集{E}的闭包 $\{E\}^+ = \{E\}$ ，闭包中无不属于子集{E}的“新属性”，所以找不到新依赖。

六、R已达3NF，是否已达BCNF？若不是BCNF，将其分解为BCNF模式集。

2. 证明R1满足BCNF：

对于一个新关系，不知道它的依赖集，不知道它的候选关键字，我们需要借助原R(A, B, C, D, E)的依赖集 $F = \{AB \rightarrow C, AC \rightarrow E, AB \rightarrow D\}$ 。

① 对R1(A, C, E)：

子集{A, C}的闭包 $\{A, C\}^+ = \{A, C, E\}$ ，闭包中有不属于子集{A, C}的“新属性”E。E属于R1，但不在子集{A, C}中，却在其闭包中出现，我们说， $\{A, C\} \rightarrow E$ 成立。

子集{A, E}的闭包 $\{A, E\}^+ = \{A, E\}$ ，闭包中无不属于子集{A, E}的“新属性”，所以找不到新依赖。

六、R已达3NF，是否已达BCNF？若不是BCNF，将其分解为BCNF模式集。

2. 证明R1满足BCNF：

对于一个新关系，不知道它的依赖集，不知道它的候选关键字，我们需要借助原R(A, B, C, D, E)的依赖集 $F = \{AB \rightarrow C, AC \rightarrow E, AB \rightarrow D\}$ 。

① 对R1(A, C, E)：

子集{C, E}的闭包 $\{C, E\}^+ = \{C, E\}$ ，闭包中无不属于子集{C, E}的“新属性”，所以找不到新依赖。

最后，关系模式R1(A, C, E)的函数依赖集 $F = \{AC \rightarrow E\}$ ，因此，关系模式R1(A, C, E)是BCNF。

六、R已达3NF，是否已达BCNF？若不是BCNF，将其分解为BCNF模式集。

3. 判断R2是否满足BCNF：

对于R2 (A, B, C, D)：

首先，写出R2的所有子集是： $\{A\}$ 、 $\{B\}$ 、 $\{C\}$ 、 $\{D\}$ 、 $\{A, B\}$ 、 $\{A, C\}$ 、 $\{A, D\}$ 、 $\{B, C\}$ 、 $\{B, D\}$ 、 $\{C, D\}$

在 $F = \{AB \rightarrow C, AC \rightarrow E, AB \rightarrow D\}$ 中：

子集 $\{A\}$ 的闭包 $\{A\}^+ = \{A\}$ ，闭包中无不属于子集 $\{A\}$ 的“新属性”，所以找不到新依赖。

同理，子集 $\{B\}$ 、 $\{C\}$ 、 $\{D\}$ 也不“带来”新依赖。

子集 $\{A, B\}$ 的闭包 $\{A, B\}^+ = \{A, B, C, D\}$ ，闭包中的“新属性”是C和D。C、D在R2中，但不在子集 $\{A, B\}$ 中，
所以有新依赖 $AB \rightarrow C$ 和 $AB \rightarrow D$ 。

六、R已达3NF，是否已达BCNF？若不是BCNF，将其分解为BCNF模式集。

3. 判断R2是否满足BCNF：

子集{A, C}的闭包 $\{A, C\}^+ = \{A, C\}$ ，无新依赖产生。

子集{A, D}的闭包 $\{A, D\}^+ = \{A, D\}$ ，无新依赖产生。

子集{B, C}的闭包 $\{B, C\}^+ = \{B, C\}$ ，无新依赖产生。

子集{B, D}的闭包 $\{B, D\}^+ = \{B, D\}$ ，无新依赖产生。

子集{C, D}的闭包 $\{C, D\}^+ = \{C, D\}$ ，无新依赖产生。

最后，关系模式R2(A, B, C, D)的函数依赖集 $F = \{AB \rightarrow C, AB \rightarrow D\}$ 。

经计算， $F = \{AB \rightarrow C, AB \rightarrow D\}$ 已是最小函数依赖集。

经计算，关系模式R2(A, B, C, D)的唯一候选关键字是{A, B}。

可以看出，关系模式R2(A, B, C, D)满足BCNF。

总结：

初步得出关系模式规范化的基本方法是：

1. 确定数据依赖，对关系模式的各个属性按数据分析阶段所得到的语义写出其数据依赖，得到一组或全部数据依赖。
2. 按照数据依赖的理论，逐一分析这组关系模式，确定属于第几范式，进行模式分解。

规范化理论致力于解决关系模式中不合适的数据依赖问题，使关系结构合理，便于操作，为用户开发创建一个良好的数据库应用系统打下了基础。

第6章 数据库设计理论

理论：概念、性质/公理/定理/推论、证明、算法等

6.1 数据依赖理论

——关于属性之间关系的理论

6.2 关系范式理论

——关于关系模式设计的规范化形式的理论

6.3 模式分解理论(关系模式规范化方法)

——关于大模式在分解为小模式过程中的理论

6.4 小结

6.4 小结

➤ 关系模式设计过程中遵循范式原则的意义：

- ❑ 关系数据库的逻辑设计原理是以减小冗余为目标的。
- ❑ 范式是符合一定级别的关系模式的集合，能够对关系模式施加相应的数据依赖约束，已经成为衡量关系模式冗余度的重要工具。
- ❑ 属于较低级别范式的关系模式冗余度相对较大，需要对其进行分解，使其达到更高一级的范式，从而降低数据冗余，避免更新、插入和删除异常，这种过程称为规范化。
- ❑ 规范化需要进行关系的模式分解，评判准则是无损连接性和保持函数依赖，分解必须具有无损连接性而尽量保持函数依赖。

6.4 小结

➤ 关系模式设计过程中遵循范式原则的意义：

▣ 一个较大关系被分解为若干个较小关系时：

- ✓ 如果分解具有无损连接性，则说明将所有较小关系自然连接可以还原为较大关系，信息不会丢失。
- ✓ 如果分解能够保持函数依赖，则说明更新任意一个较小关系时，为了判断更新数据是否遵从原有的函数依赖，只需要考虑该较小关系的函数依赖，而不必连接多个较小关系

6.4 小结

➤ 遵循范式原则和适度冗余之间的折中：

- ❑ 遵循关系范式原则，则需要将一个关系模式，拆解成两个或多个小的模式；
- ❑ 而查询时，需要将这两个或多个小的模式联结成一个模式；
- ❑ 遵循关系范式原则避免了冗余、插入异常、删除异常等问题。
- ❑ 但由于联结运算的低效率，使得查询速度很慢。
- ❑ 因此需要折中。

数据库系统



➤ 求解候选码基本算法的具体步骤(方法一)

- 第1步：求关系模式 $R\langle U, F \rangle$ 的最小函数依赖集 F ;
- 第2步：将属性分类为 UL, UR, UB ;
- 第3步：计算候选码
 - ✓若 $UL \neq \Phi$, 则其中的所有属性都是主属性, 计算 UL 的闭包
 - 若 $UL^+ = U$, 则 UL 为 R 的唯一的候选码;
 - 若 $UL^+ \neq U$, 转第4步。
 - ✓若 $UL = \Phi$, 转第4步;
- 第4步：将 UL 依次与 UB 中的属性组合, 并分别计算闭包 UL^+
 - ✓若 $UL^+ = U$, 则 UL 中的属性集就是一个候选码;
 - ✓若 $UL^+ \neq U$, 转第4步, 直至 UB 中的所有属性处理完。

简而言之：取最小依赖集，计算 UL 闭包，如果 UL 闭包包含全属性，则 UL 为唯一候选码，如果不包含，则依次与 UB 属性组合后再求闭包是否包含全属性。



➤ 最小函数依赖集算法

- ❑ 第一步:使F中的任何一个函数依赖的右部仅含有一个属性;
- ❑ 第二步:去掉多余的函数依赖;
 - 从第一个函数依赖 $X \rightarrow Y$ 开始将其从F中去掉,然后在剩下的函数依赖中求X的闭包 X^+ ,看 X^+ 是否包含Y,若是,则去掉 $X \rightarrow Y$;否则不能去掉,依次做下去。直到找不到冗余的函数依赖;
- ❑ 第三步:去掉各依赖左部多余的属性;
 - 一个一个地检查函数依赖左部非单个属性的依赖。

➤ 求解候选码基本算法的具体步骤(方法三)

对于给定的 $R(U)$ 和函数依赖集 F , 可以将它的属性划分为4类:

L类、R类、N类、LR类

- 定理1: 对于给定的关系模式 R 及其函数依赖集 F , 若 $X(X \in R)$ 是L类属性, 则 X 必为 R 的任一候选码的成员。
- 推论1: 对于给定的关系模式 R 及其函数依赖集 F , 若 $X(X \in R)$ 是L类属性, 且 X^+ 包含了 R 的全部属性, 则 X 必为 R 的唯一候选码。
- 定理2: 对于给定的关系模式 R 及其函数依赖集 F , 若 $X(X \in R)$ 是R类属性, 则 X 不在任何候选码中。
- 定理3: 设有关系模式 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2: 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 的所有属性, 则 X 是 R 的唯一候选码。

L、R、N、LR类:

- 根据定理, L、N类必为候选码之一;
- 如果 L^+ 包含全部 R , 则 L 为唯一候选码。
- R类不在任何候选码中。
- $L+N$ 类且 $(L+N)^+$ 包含所有 R , 则 $(L+N)$ 为唯一候选码。

[返回](#)