

第6章 关系数据库设计理论

6.1 问题提出

6.1.1 关系模式: $R(U, D, DOM, F)$

U: 属性全集

D: U中属性域

DOM: 属性向域的映像集合

F: 属性间数据的依赖关系集

简化关系模式表示: $R(U, F)$

6.1.2 数据依赖

——属性间的依赖关系

Student (SNO, NM, SEX, BYR)

$SNO \rightarrow NM, SNO \rightarrow SEX, (SNO, NM) \rightarrow BYR,$

$BYR \nrightarrow SEX$

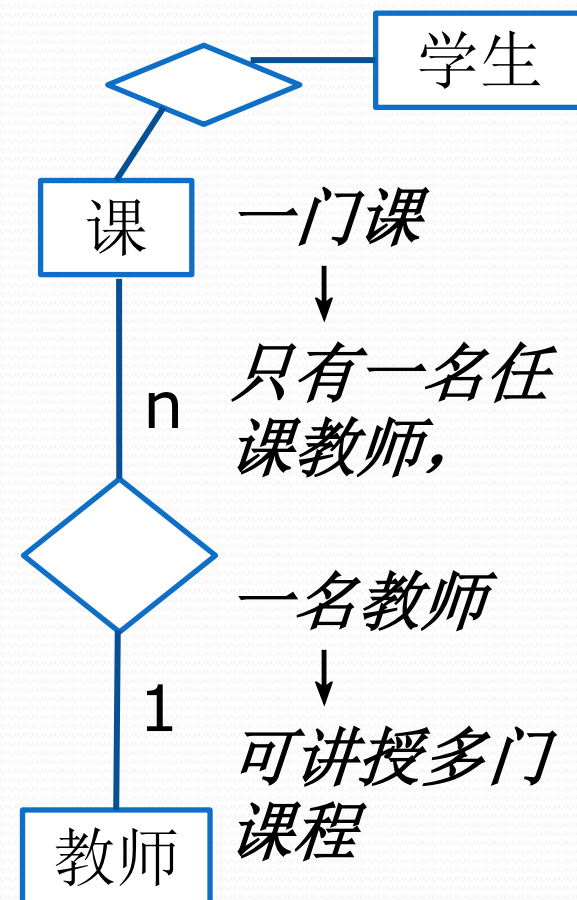
关系内部属性之间的一种约束, 通过值的相等与否体现, 是现实世界属性间联系的抽象, 是语义的体现。



6.1.3 R(U, F)存在问题

R_1 : NM: 教师姓名, ADDR: 教师地址

SNO	CNO	CNM	TNM	ADDR	GRADE
981	C ₁	OS	玉迷	D ₁	70
982	C ₂	DB	范思	D ₂	72
982	C ₄	AI	范思	D ₂	72
983	C ₁	OS	玉迷	D ₁	70
982	C ₁	OS	玉迷	D ₁	85



结构是否合理?

问题存在于何处? “更新”操作

增、删、改

1. 冗余大

多人选相同的课程，CNO，CNM，**TNM**，**ADDR**冗余

2. 修改麻烦

冗余直接导致某门课程换教师时的修改麻烦

3. 插入操作异常(insertion anomalies)

——需要插入到DB中的数据插入不进去，例如**课程**、**教师**信息。
因为——候选码：(SNO, CNO), (SNO, CNM)

4. 删除异常(deletion anomalies)

——不应该删去的信息丢失了。

例如可能出现的情况：删去某些学生选课数据，导致**课程及教师**信息丢失。

【综上所述】：对“**操作异常**”的分析

6.1.4 结论

- 1.一个好的关系模式应冗余尽可能少;
- 2.一个好的关系模式应避免插入、删除异常;
- 3.原因是关系模式中**存在不合适的属性间联系**;
- 4.解决方法是消去不合适联系;
- 5.采用分解的策略消去。

6.2 规范化

6.2.1 函数依赖概念(functional dependency)

1. 定义:

设有关系模式 $R(U, F)$, X 、 Y 是 U 的子集。对于 $R(U)$ 的任何一个具体关系值集 r , t 、 s 是 r 中的任意两个元组。若有 $t[x]=s[x] \Rightarrow t[y]=s[y]$, 则称 $x \rightarrow y$ 。

(任给一 x , 有唯一 y 与之对应, 则 $x \rightarrow y$)

注: “ $x \rightarrow y$ ”与 F 的关系。



上例中：

$CNO \rightarrow ADDR$, $CNM \rightarrow TNM$, $CNM \rightarrow ADDR$, $TNM \rightarrow ADDR$
 $CNO \not\rightarrow GRADE$, $SNO \not\rightarrow GRADE$, $SNO \not\rightarrow ADDR$,
 $(SNO, CNO) \rightarrow GRADE$

2. 说明

- 1) 依据定义判断R中函数依赖（fd）必须分析其全部元组；
- 2) 有时可依据客观语义确定函数依赖；

例如student表中：

$SNO \rightarrow NM$ $NM \xrightarrow{?} BYR$

- 3) 1:1, m:1存在函数依赖；
- 4) 1:m, m:n不存在函数依赖。

平凡函数依赖

1. 定义

设有关系模式 (U, F) , x 、 y 都是 U 的子集, 若 $x \rightarrow y$, $y \subseteq x$, 则称 $x \rightarrow y$ 是平凡函数依赖。

$(SNO, CNO) \rightarrow CNO$

$CNO \subseteq (SNO, CNO)$

2. 说明

对任何 R , 平凡函数依赖总是成立的。

非平凡函数依赖

1. 定义

设有关系模式 (U, F) , x 、 y 是 U 的子集, 如果 $x \rightarrow y$, 但 $y \not\subseteq x$, 则称 $x \rightarrow y$ 是非平凡函数依赖。

$CNO \rightarrow NM$ $NM \not\subseteq CNO$, 非平凡函数依赖

$(SNO, CNO) \rightarrow \text{GRADE}$ $\text{GRADE} \not\subseteq (SNO, CNO)$

2. 说明

非平凡函数依赖

若未特别声明, 今后总是讨论非平凡函数依赖。

完全函数依赖(fulley fd)

定义：设有关系模式 $R(U, F)$ ， x 、 y 是 U 的子集，若 $x \rightarrow y$ ，且对于 x 的任何真子集 x' ($x' \subset x$)，都有 $x' \not\rightarrow y$ ，则称 y 完全函数依赖于 x ，记作 $x \xrightarrow{f} y$ 。

$(SNO, CNO) \xrightarrow{f} GRADE \quad \because SNO \not\rightarrow GRADE,$
 $CNO \not\rightarrow GRADE$

部分函数依赖(Partional fd)

定义：设有关系模式 $R(U, F)$ ， x 、 y 是 U 的子集，若 $x \rightarrow y$ ，且存在一个 x 的真子集 x' ($x' \subset x$)，使得 $x' \rightarrow y$ ，则称 y 部分函数依赖于 x ，记作 $x \xrightarrow{P} y$

$(SNO, CNO) \rightarrow NM$

$\because CNO \rightarrow NM,$

$\therefore (SNO, CNO) \xrightarrow{P} NM$

传递fd (transitive fd)

1. 定义

任给关系模式 $R(U, F)$, x, y 是 U 的子集, 若 $x \rightarrow y, y \rightarrow z$, 且 $y \not\rightarrow x, y \not\subseteq x, z \not\subseteq y$, 则称 z 传递函数依赖于 x , 记作 $x \xrightarrow{t} z$

$CNM \rightarrow TNM, TNM \rightarrow ADDR, TNM \not\rightarrow CNM,$
 $\therefore CNM \xrightarrow{t} ADDR$

双重
影响

2. 说明

加上 $y \not\rightarrow x$, 是避免 $x \leftrightarrow y$, 否则 $x \xrightarrow{\text{直接}} z$



6.2.2 码

1. 码的形式化定义

定义: 设有关系模式 $R(U, F)$, X 为 U 的子集, 若 $X \xrightarrow{f} U$, 则 x 为 R 的一个候选码(Candidate Key)。

R1中:

$(SNO, CNO) \rightarrow (SNO, CNO, CNM, TNM, ADDR, GRADE)$
 $(SNO, CNM) \rightarrow (SNO, CNO, CNM, TNM, ADDR, GRADE)$

2. 主码 (Primary Key)

定义：任选候选码之一。

3. 主属性

定义：属于任何一个候选码中的属性称为主属性

R1中：SNO, CNO, CNM为主属性。

4. 非主属性

定义：不属于任何候选码中的属性称为非主属性。

R1中：TNM, ADDR, GRADE为非主属性。

特例：整个属性组是码，则称为全码。

6.2.3 范式(Relation Normalization)

1. 关系规范化(Relation Normalization)

——将关系模式从低级范式向高级范式分解的过程。

2. 范式(Normal Form)

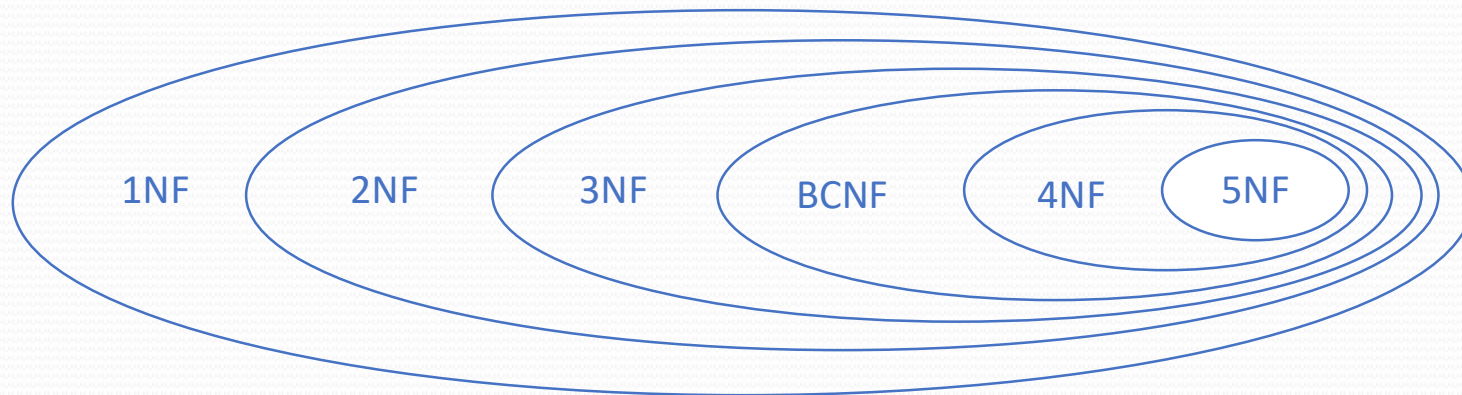
——满足某种条件规范的关系模式集合。

3. 级别

1NF→2NF→3NF→BCNF→4NF→5NF。

范式的包含关系

一个关系中不出
现某种类型的数
据依赖

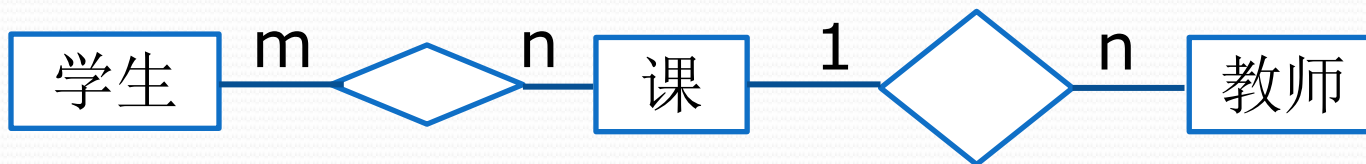


4. 目标

概念单一化：一个关系模式表示一个概念，一个实体，**一个实体间联系**；多余部分分解出去。



尽量不要让一个属性受到除主属性外其他属性的影响



1NF (First Normal Form)

1. 定义

任给关系模式 $R(U, F)$ ，若 U 中每个属性均为不可再分的基本数据元素（原子项），则 $R \in 1NF$ 。

2. 说明

1) 关系数据库系统中，所有关系模式都必须是1NF。

2) $R1 \in 1NF$ 。

3) 下面 R 为非1NF（非规范关系模式）：

R

SNO	CNO	Grade		
		test1	test2	test3
.....

3. 转换

将非1NF \Rightarrow 1NF

R				
SNO	CNO	test1	test2	Test3

例：嵌套属性

扣税记录



交易		商场	是否免税	税金
顾客	商品			



顾客	商品	商场	是否免税	税金

4. 存在问题

如R为1NF，可能存在冗余，修改麻烦，操作异常问题。

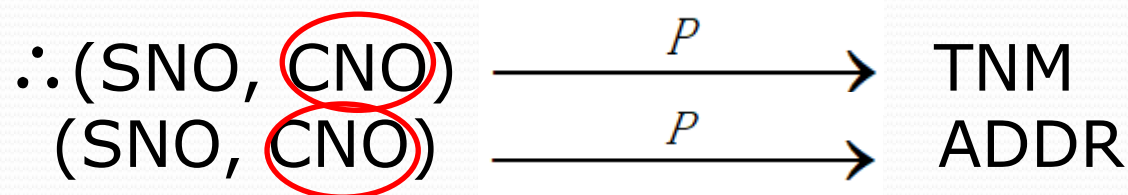
5. 原因：

——非主属性部分fd于候选码。

R中：

$\because \text{CNO} | \text{CNM} \rightarrow \text{NM}$

$\text{CNO} / \text{CNM} \rightarrow \text{ADDR}$



6. 投影分解

——消去非主属性对候选码的部分fd。

1) 将满足完全fd的属性集分解组成一个关系模式R2。

2) 将导致部分fd的属性集分解成一个关系模式R3。

R2

SNO	CNO	GRADE
981	C1	70
982	C2	72
982	C4	72
983	C1	70
982	C1	85

R3

CNO	CNM	TNM	ADDR
C1	OS	玉迷	D1
C2	DB	范思	D2
C4	AI	范思	D2

依据理论分析
人工解决问题

7. 效果

1) 冗余减少

仍然存在多个CNO（三个分量值）；

选同一课程的CNM, TNM, ADDR仅出现一次。

2) 避免了原修改麻烦

OS换教师、修改R₃中一个元组。

3) 避免了原插入操作异常。

新开课程无人选修仍可插入到DB中（R₃）。

4) 避免了原删除异常

删除学生信息在R₂中进行，R₃中的课程、教师数据不受影响。

R2

SNO	CNO	GRADE
981	C1	70
982	C2	72
982	C4	72
983	C1	70
982	C1	85

R3

CNO	CNM	TNM	ADDR
C1	OS	玉迷	D1
C2	DB	范思	D2
C4	AI	范思	D2

6.2.4 2NF

1. 定义

任给 $R(U, F)$ ，若 $R \in 1NF$ ，且每个非主属性都完全函数依赖于候选码，则 $R \in 2NF$ 。

换言之：2NF不允许 F 中有这样的函数依赖“ $X \rightarrow Y$ ”，其中 X 是码的真子集， Y 是非主属性。

1) R_2

候选码: (SNO, CNO)

非主属性: GRADE

fd: $(SNO, CNO) \xrightarrow{f} GRADE$ $\therefore R_2 \in 2NF$

R2

SNO	CNO	GRADE
-----	-----	-------

2) R_3 中:

候选码: CNO, CNM

非主属性: TNM, ADDR

fd: $CNO \xrightarrow{f} TNM$, $CNO \xrightarrow{f} ADDR$, $CNM \xrightarrow{f} TNM$,
 $CNM \xrightarrow{f} ADDR$

R3

CNO	CNM	TNM	ADDR
-----	-----	-----	------

 \therefore 根据定义: $R_3 \in 2NF$

例: 关系模式 $R(A, B, C, D)$, 码为 AB, 给出它的一个函数依赖集, 使得 R 属于 1NF 而不属于 2NF

答案: $AB \rightarrow CD$, $B \rightarrow C$ 答案之一, $B \rightarrow D$, $A \rightarrow D$...

2. 问题

1) 冗余仍存在

R2: 同一课程多人选修, CNO冗余(必要冗余)。

R3: 一教师讲授多门课程, TNM, ADDR冗余。

2) 修改麻烦

教师搬家, 修改多个元组。

3) 插入操作异常

新教师报到, 未承担任何教学任务, 教师信息不能进入DB。

4) 删除异常

删去课程信息, 同时会删去教师信息。

R2

SNO	CNO	GRADE
-----	-----	-------

教师的基本信息与任课关系并存于一张表, 主题不单一!!!

R3

CNO	CNM	TNM	ADDR
-----	-----	-----	------

3. 原因

——非主属性对候选码的传递fd。

R3

R3中:

CNO	CNM	TNM	ADDR
-----	-----	-----	------

候选码: CNO, CNM

非主属性: TNM, ADDR

fd: $CNO \rightarrow TNM$, $TNM \rightarrow ADDR$, $TNM \not\rightarrow CNO$;

$\therefore CNO \xrightarrow{t} ADDR$

4. 投影分解

——消去非主属性对码的传递fd。

——策略是使“传递”的动作不在表内发生。



R4

CNO	CNM	TNM
C1	OS	玉迷
C2	DB	范思
C4	AI	范思

R5

TNM	ADDR
玉迷	D1
范思	D2

5. 效果

1) 冗余有条件地减少

一个教师讲授课程多时，冗余减少（如一个教师讲4门课，则减少3个ADDR分量值）。

2) 避免了原修改麻烦（地址）

3) 避免了原插入异常（教师）

4) 避免了原删除异常（课程）

R4

CNO	CNM	TNM
C1	OS	玉迷
C2	DB	范思
C4	AI	范思

R5

TNM	ADDR
玉迷	D1
范思	D2

6.2.5 3NF

1. 定义

任给R (U, F)，若不存在这样的码X、属性组Y及非主属性Z（Z不包含于Y），使得 $X \longrightarrow Y$ ， $(Y \not\rightarrow X)$ ， $Y \longrightarrow Z$ 成立，则 $R \in 3NF$ 。

传递？

比传递
范围更广

传递fd (transitive fd)

1. 定义

任给关系模式R(U, F)，x、y是U的子集，若 $x \rightarrow y$ ， $y \rightarrow z$ ，且 $y \not\rightarrow x$ ， $y \not\subseteq x$ ， $z \not\subseteq y$ ，则称z传递函数依赖于x，记作 $x \xrightarrow{t} z$ 。

任给R (U, F)，若R为2NF，且其每一个非主属性都不传递fd于候选码，则 $R \in 3NF$ 。

理解：3NF不允许F中有这样的非平凡函数依赖“ $X \longrightarrow Y$ ”，其中X不包含码，Y是非主属性。

1) R_4 .

候选码: CNO, CNM

非主属性: TNM

∵ 每个候选码均为单个属性, 显然不存在非主属性对候选码的部分函数依赖。

∴ $R_4 \in 2NF$.

又 ∵ 只存在一个非主属性, 不存在该非主属性对候选码的传递函数依赖。

∴ $R_4 \in 3NF$

2) R_5

∵ R_5 中只存在一个非主属性和一个主属性, 显然为 **2NF**, 同时又满足 **3NF** 定义。

∴ $R_5 \in 3NF$

R4

CNO	CNM	TNM
-----	-----	-----

E?/R?

R5

TNM	ADDR
-----	------

推论: 任何二元关系模式必为**3NF**。

例：关系模式R (A, B, C, D)，码为AB，给出它的一个函数依赖集，使得R属于2NF而不属于3NF

答案： $AB \rightarrow CD, C \rightarrow D$ ；或者 $AB \rightarrow CD, BC \rightarrow D$

2. 分析评价

- 1) 部分fd和传递fd是冗余及操作异常的重要根源。
- 2) 3NF不存在非主属性对候选码的部分fd和传递fd。
- 3) 3NF消去了大部分冗余及操作异常。
- 4) 并非所有的3NF都能完全消除冗余及操作异常。

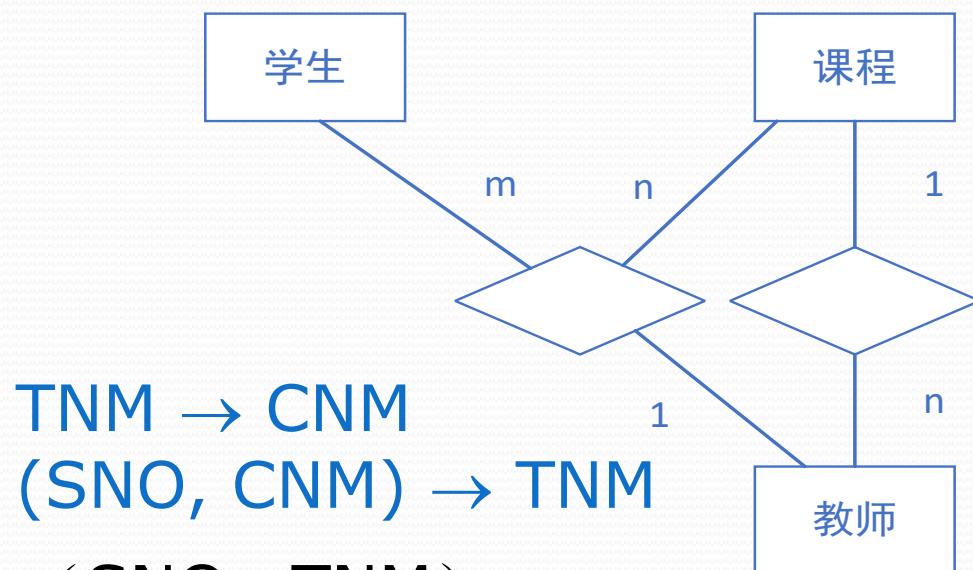
$AB \rightarrow CD, CD \rightarrow A?$ 或者 $AB \rightarrow CD, C \rightarrow A?$

再来看一个例子

R6

SNO	CNM	TNM
S1	OS	擎武
S2	OS	菲扬
S4	DB	阿泰
S5	DB	阿泰
S5	OS	擎武
...

每一教师只讲一门课，
每门课可由多个教师讲，
学生一旦选定一门课，就确定了
其任课教师



候选码：(SNO, CNM)，(SNO, TNM)

非主属性：无

fd：无非主属性对候选码的部分fd和传递fd。

$\therefore R6 \in 3NF$

3. 3NF存在问题（可能存在）

1) 冗余

多个学生选同一个教师的同一门课时，CNM重复。

2) 修改麻烦

同一教师讲授的课程改名，则修改多个元组。

3) 插入异常

(SNO, CNM), (SNO, TNM)

教师开设某门课程，尚无学生选修，无SNO，则课程、教师信息不能进入DB。

4) 删除异常

删去学生信息丢失课程及教师信息。

4. 原因

①存在主属性对候选码的部分fd。

R6中：

$\because \text{TNM} \rightarrow \text{CNM}$

$\therefore (\text{SNO}, \text{TNM}) \xrightarrow{p} \text{CNM}$

② 存在主属性对候选码的传递fd。

$\because (\text{SNO}, \text{CNM}) \rightarrow \text{TNM}$

$\text{TNM} \rightarrow \text{CNM}$

$\therefore (\text{SNO}, \text{CNM}) \xrightarrow{t} \text{CNM}$

5. 投影分解

——消去主属性对候选码的部分及传递fd。

将R6分解

R6

SNO	CNM	TNM
S1	OS	擎武
S2	OS	菲扬
S4	DB	阿泰
S5	DB	阿泰
S5	OS	擎武
...

将R6分解为:

R7

SNO	TNM
S1	擎武
S2	菲扬
...	...

R8

TNM	CNM
擎武	OS
菲扬	OS
阿泰	DB
...	...

分解为(SNO,CNM),
(TNM,CNM)?

再补充(SNO,TNM)?

6. 效果

1、冗余得到了较好控制

多了NM，但很多学生选同一教师的一门课程时，课程信息只存一次。

2、修改麻烦避免了一名教师的课程改名后，只在R8中改一元组。

3、插入异常避免了R8中可以插入无学生选修的课程和教师信息。

4、删除异常避免了删去学生信息在R7中进行，教师及课程信息在R8中不受影响。

R7

SNO	TNM
S1	擎武
S2	菲扬
...	...

R8

TNM	CNM
擎武	OS
菲扬	OS
阿泰	DB
...	...

6.2.6 BCNF (Boyce Codd Normal Form)

1、定义

任给 $R(U, F)$ ， X 、 Y 为 U 中属性子集，若 $R \in 1NF$ ，且对 R 的每一个 $X \rightarrow Y$ ， $Y \not\subseteq X$ ， X 必包含候选码，则 $R \in BCNF$ 。

换言之：若 R 中的每一个函数依赖中的左部决定属性集都包含有候选码，则 $R \in BCNF$ 。

1) R7

(SNO, NM) 为候选码，无特殊的函数依赖， $\therefore R7 \in BCNF$

2) R8

$TNM \rightarrow CNM$ ， TNM 为候选码； $\therefore R8 \in BCNF$

2元关系都是
BCNF

2、BCNF性质

- 1) 所有非主属性都完全函数依赖于候选码；
- 2) 所有非主属性都不传递函数依赖于候选码；
- 3) 所有主属性都完全函数依赖于不包含它的候选码；
- 4) 所有主属性都不传递函数依赖于候选KEY。

没有任何属性函数依赖于非码的任何一组属性。

3、定理：如果 $R \in \text{BCNF}$ ，则 $R \in 3\text{NF}$

证（反证法）

理解：3NF不允许F中有这样的非平凡函数依赖“ $X \rightarrow Y$ ”，其中X不包含码，Y是非主属性。

设 $R \in \text{BCNF}$ ，但 $R \notin 3\text{NF}$ ，则总可找到属性集 x, y, z ，其中 z 为非主属性， y 不包含候选码， x 为候选码（ $R \notin 3\text{NF}$ ，则存在非主属性，它部分或者传递函数依赖于码），从而它们之间存在违反3NF定义的函数依赖的传递，即 $y \rightarrow z$ 成立，且其左边的决定因素不含候选码。

根据BCNF定义， $R \notin \text{BCNF}$ ，与假设矛盾。

定理得证。

3NF定义

任给 $R(U, F)$ ，若不存在这样的码 X 、属性组 Y 及非主属性 Z （ Z 不包含于 Y ），使得 $X \rightarrow Y$ ， $(Y \not\rightarrow X)$ ， $Y \rightarrow Z$ 成立，则 $R \in 3\text{NF}$ 。

例：关系模式SCO(S#, C#, O)，表示学生选修课程的名次，有函数依赖(S#, C#)→ O, (C#, O) → S#，它属于3NF、BCNF吗？

答案：

- (S#,C#)或者(C#,O)都可以作为候选码
- 不存在非主属性对码传递依赖或部分依赖，SCO ∈ 3NF
- 没有其他决定因素，SCO ∈ BCNF

问题判定的关键环节：

分析属性间的依赖关联，找出关系的**候选码**。

6.2 总结

1、关系规范化进程

非规范关系

去掉嵌套属性、重复组

1NF

消去非主属性对候选KEY的部分fd

2NF

进一步消去非主属性对候选KEY的传递fd

3NF

进一步消去主属性对候选KEY的部分和传递fd

BCNF

2、结论

- 1) 3NF必定为2NF和1NF，反之不一定；
- 2) BCNF必为3NF，反之不一定；
- 3) 3NF已在很大程度上控制了数据冗余；
- 4) 3NF已在很大程度上消去了插入和删除操作异常；
- 5) 3NF分解仍不够彻底（可能存在主属性对候选码的部分fd和传递fd）；
- 6) 在函数依赖范围内，BCNF已完全消去了插入删除异常；
- 7) 范式并非越高越好；
- 8) 依然存在其它问题（冗余垃圾，连接运算）；
- 9) 分解不唯一。



分解不唯一

R7

SNO	TNM
S1	擎武
S2	菲扬
...	...

TNM	CNM
擎武	OS
菲扬	OS
阿泰	DB
...	...

R9

SNO	CNM
S1	OS
S2	OS
...	...

R10

TNM	CNM
擎武	OS
菲扬	OS
阿泰	DB
...	...

新问题

例:R

CNM	TNM	BOOK
数学	星驰	数学分析
数学	星驰	高等代数
数学	星驰	微分方程
数学	孟达	数学分析
数学	孟达	高等代数
数学	孟达	微分方程

CNM: 课程名

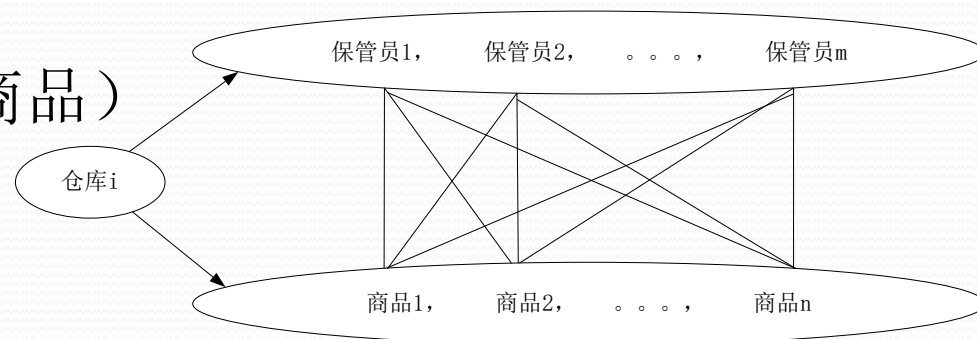
TNM: 教师名

BOOK: 参考书名

物理	小刚	普通物理学
物理	小刚	光学原理
物理	艺谋	普通物理学
物理	艺谋	光学原理
物理	凯歌	普通物理学
物理	凯歌	光学原理
.....

教材例6.10:

关系（仓库，保管员，商品）



6.4 多值依赖 (MVD: multivalued dependency)

(本节内容作为课外阅读)

6.3 数据依赖的公理系统

函数依赖的公理系统——Armstrong公理系统（一个有效而完备的公理系统）



求关系的候选码的算法；
实现关系规范化分解算法

一切，从函数依赖说起

函数依赖集就是表象上的F?

No! 有内涵!

● 逻辑蕴涵

定义:对于满足一组函数依赖F的关系模式R $\langle U, F \rangle$, 其任何一个关系r, 若函数依赖 $X \rightarrow Y$ 都成立(即r中任意两元组t、s, 若 $t[X]=s[X]$, 则 $t[Y]=s[Y]$), 则称F逻辑蕴含 $X \rightarrow Y$

(记为 $F \vdash X \rightarrow Y$)

作用: 确定关系的候选码、确定关系的范式级别、关系的分解正确性判断时需要对F逻辑蕴涵的函数依赖判断。

● F的闭包—— F^+

定义: 被F所逻辑蕴涵的函数依赖的全体所构成的集合称作F的闭包, 记作 $F^+ = \{X \rightarrow Y \mid F \vdash X \rightarrow Y\}$

F的闭包

n个属性:

$O(2^{2n})$

$F = \{X \rightarrow Y, Y \rightarrow Z\}$, F^+ 验证是指数级的计算复杂度,

$F^+ = ? \{$

$X \rightarrow \varnothing, Y \rightarrow \varnothing, Z \rightarrow \varnothing,$	$XY \rightarrow \varnothing,$	$XZ \rightarrow \varnothing, YZ \rightarrow \varnothing,$	$XYZ \rightarrow \varnothing,$
$X \rightarrow X, Y \rightarrow Y, Z \rightarrow Z,$	$XY \rightarrow X,$	$XZ \rightarrow X, YZ \rightarrow Y,$	$XYZ \rightarrow X,$
$X \rightarrow Y, Y \rightarrow Z,$	$XY \rightarrow Y,$	$XZ \rightarrow Y, YZ \rightarrow Z,$	$XYZ \rightarrow Y,$
$X \rightarrow Z, Y \rightarrow YZ,$	$XY \rightarrow Z,$	$XZ \rightarrow Z, YZ \rightarrow YZ,$	$XYZ \rightarrow Z,$
$X \rightarrow XY, \quad . \quad . \quad .$	$XY \rightarrow XY, \quad XZ \rightarrow XY,$		$XYZ \rightarrow XY,$
$X \rightarrow XZ, \quad . \quad . \quad .$	$XY \rightarrow YZ, \quad XZ \rightarrow XZ,$		$XYZ \rightarrow YZ$
$X \rightarrow YZ, \quad . \quad . \quad .$	$XY \rightarrow XZ, \quad XZ \rightarrow XY,$		$XYZ \rightarrow XZ,$
$X \rightarrow ZYZ, \quad . \quad . \quad .$	$XY \rightarrow XYZ, XZ \rightarrow XYZ,$		$XYZ \rightarrow XYZ \}$

问题

根据定义，只能逐个的判断某个函数依赖是否被关系的值满足，从而判断该函数依赖是否被F逻辑蕴涵（该函数依赖属于 F^+ ），提出了一个“放之四海皆需准”的标准的验证问题，“举轻若重”，具体验证操作成本高。

人们更愿意“四两拨千斤”，思路有哪些？

- 运用推理规则，三点确定一个平面、一个系统。。。
- 正面不易说清楚，则看看反面，试试反证法通过反例推出矛盾。

- 如何能够不依据定义获取F的逻辑蕴含闭包？
建立一套推理规则，通过算法求出（或可校验）F所蕴涵的各个函数依赖。

- 解决者：Armstrong

相应的一套推理规则被命名为Armstrong公理系统。

- Armstrong公理系统：

关系模式 $R\langle U, F \rangle$ ， U 为属性集， F 是 U 上的一组函数依赖，存在下述三条 F 上的推理规则

- 自反律(reflexivity): 若 $Y \subseteq X \subseteq U$ ，则 $X \rightarrow Y$
- 增广律(augmentation): 若 $X \rightarrow Y$ ， $Z \subseteq U$ ，则 $XZ \rightarrow YZ$
- 传递律(transitivity): 若 $X \rightarrow Y$ ， $Y \rightarrow Z$ ，则 $X \rightarrow Z$

● 对其扩充——由Armstrong公理导出的推理规则

● 合并规则(union rule)

若 $X \rightarrow Y$, $X \rightarrow Z$, 则 $X \rightarrow YZ$

● 分解规则(decomposition rule)

若 $X \rightarrow YZ$, 则 $X \rightarrow Y$, $X \rightarrow Z$

● 伪传递规则(pseudo transitivity rule)

若 $X \rightarrow Y$, $WY \rightarrow Z$, 则 $WX \rightarrow Z$



能用么？

使用Armstrong公理系统及其扩充推理规则的前提条件：
——“Armstrong公理系统必须是有效的、完备的”

- 有效性**：用Armstrong公理从F中导出的函数依赖**必为F所蕴涵**。（相当于公理系统的正确性）
- 完备性**：F所蕴涵的函数依赖都**能用Armstrong公理从F中导出**。

即 “被F所蕴涵”  “可由Armstrong公理导出”

F的闭包—— F^+ （F逻辑蕴涵的函数依赖集合）



{F出发借助Armstrong公理导出的函数依赖}

问题：如何证明？

答：有效性/正确性（易证），证明完备性——**构造法+反证法**

1) 自反律 (若 $Y \subseteq X \subseteq U$, 则 $X \rightarrow Y$) :

证明: $R\langle U, F \rangle$ 上的任一关系 r , 其中任意两个元组 $t, s \in r$,
如果 $t[X] = s[X]$, 且 $Y \subseteq X$, 则显然 $t[Y] = s[Y]$, 满足 $X \rightarrow Y$
被 F 逻辑蕴涵的定义, 故自反律是正确的。

2) 增广律 (若 $X \rightarrow Y$, $Z \subseteq U$, 则 $XZ \rightarrow YZ$) :

证明: $R\langle U, F \rangle$ 上的任一关系 r , 其中任意两个元组 $t, s \in r$,
若 $t[XZ] = s[XZ]$, 则有 $t[X] = s[X]$ 和 $t[Z] = s[Z]$ 。
又因为 $X \rightarrow Y$, 所以 $t[Y] = s[Y]$, 所以 $t[YZ] = s[YZ]$ 。
故 $XZ \rightarrow YZ$ 被 F 逻辑蕴涵。

3) 传递律 (若 $X \rightarrow Y$, $Y \rightarrow Z$, 则 $X \rightarrow Z$) ;

证明: $R\langle U, F \rangle$ 上的任一关系 r , 其中任意两个元组 $t, s \in r$,
若 $t[X] = s[X]$, 则由 $X \rightarrow Y$ 可知 $t[Y] = s[Y]$, 同理,
由 $Y \rightarrow Z$ 可知 $t[Z] = s[Z]$, 故 $X \rightarrow Z$ 被 F 逻辑蕴涵。

综上所述, Armstrong 公理系统是有效的 (正确的)。

如何证明Armstrong公理的完备性？（F所蕴涵的函数依赖都能用Armstrong公理从F中导出）

蕴含？有多少个蕴含？太多，说不清。

换个思路，逆否命题：

不能用公理推导出的就一定不被逻辑蕴含，似乎容易说清楚了。

举例：“你这个人，一怎么。。。。，就怎么。。。。”——似乎比较感性，难以接受

“我这个人，只要不怎么。。。。，就不怎么。。。。”——似乎比较理性，可接受

可是:

要判断某个函数依赖是否能由 **Armstrong** 公理导出, 有时也难以直接得出结论, 特别是当需要证明某个函数依赖不能由 **Armstrong** 公理导出时, 直观的方法需要求出这个由公理导出的函数依赖集合。

↓类似逻辑蕴含闭包不可求解的
↓问题再次出现

求函数依赖的推导集合计算量也过大 (n 个函数依赖可能推导出 2^n 个不同的函数依赖, n 过大则计算机无法处理)。

↓

↓再看 **Armstrong** 公理 (约法三章)

Armstrong 公理有效性的一个重要应用——

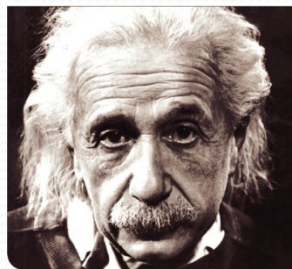
引理 6.1: $X \rightarrow A_1 A_2 \dots A_k$ 可由公理导出的充要条件是 $X \rightarrow A_i$
($i=1, 2, \dots, k$)

(推论: 合并规则和分解规则)

不妨将所有能由公理导出的 $X \rightarrow A_i$ 依赖的单个属性 A_i 定义为一个集合 X_F^+ ，如果属性集 Y 包含于该集合，即 Y 的每个属性都函数依赖于 X ，则根据引理6.1，这等价于 $X \rightarrow Y$ 可由公理导出，反之，如果 Y 不包含于这个集合 X_F^+ ，等价于 $X \rightarrow Y$ 不能由公理导出。

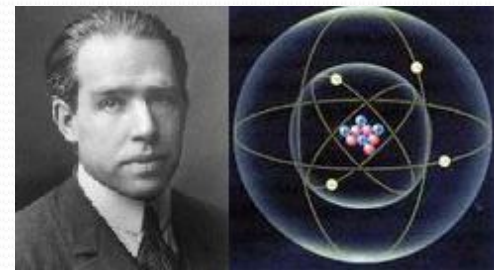
突破点：判断 $X \rightarrow Y$ 能否由公理导出不需要求得 F 导出的全部依赖的集合，只需要求得 X_F^+ 。

剩下的问题—— X_F^+ 能否被计算机在令人满意的时间(至少不超过 2^n 数量级或者 $n!$ 时间)内求出???



可以

求解空间变了



函数依赖的推导验证至此可行，如何关联到逻辑蕴含？从而分析公理的完备性？ **F 所蕴涵的每一个函数依赖都能用Armstrong公理从 F 中导出么？**（不能推导出就不被蕴含？）

构造反例，
推出矛盾（课外阅读）

首先解决属性闭包的问题

- X 关于函数依赖集 F 的闭包

定义：设 F 为属性集 U 上的一组函数依赖， $X \subseteq U$ ，

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

称 X_F^+ 为属性集 X 关于函数依赖集 F 的闭包

用途？——由属性闭包的定义可做如下判断：

- 引理6.2：设 F 为属性集 U 上的一组函数依赖， $X, Y \subseteq U$ ，则 $X \rightarrow Y$ 能由 F 导出的充要条件是 $Y \subseteq X_F^+$ 。

- 如何证明 X_F^+ 能够在足够小的时间内得到？

思路：构造一个算法，输入 X, F ，输出 X_F^+ ，如能证明算法必在足够小的时间内正确的结束，则问题得证。

● 算法 6.1 (求属性集X关于函数依赖集F的闭包)

Input: X, F

Output: X_F^+

方法: 计算 $X^{(i)}$ ($i = 1, 2, \dots, n$)

(1) $X^{(0)} := X, i = 0$

(2) 在F中寻找左边是 $X^{(i)}$ 的子集的函数依赖 $Y_j \rightarrow Z_j$ ($j = 1, 2, \dots, k$), $Y_j \subseteq X^{(i)}$,

所有 Z_j 中的属性构成集合A;

$X^{(i+1)} = X^{(i)} \cup A$;

(3) 判断是否有 $X^{(i+1)} = X^{(i)}$ 或者 $X^{(i+1)} = U$, 若是则转

(4), 否则转 (2)

(4) 输出 $X^{(i)}$, 即为 X_F^+

分析: 一旦 $X^{(i+1)} = X^{(i)}$ 则算法即使再进入(2)的循环, 输出结果也不会再改变。

算法正确性说明：

算法6.1的第(2)步执行到第 i 次，则输出结果包含所有经过 i 次推导就能导出的属性 A ，算法结束条件为结果等于 U （属性闭包不可能超过 U ）或者输出结果和上次相比没有变化（此时即使在已有结果上继续任意多次推导，输出结果也不会再增加，即能够被推导出来的属性肯定全部都在当前结果中），故算法的最终结果就是满足定义的 X_F^+ 。

算法执行的时间开销：

算法的第(2)步每执行一次，输出结果元素个数加 m （ $m \geq 1$ ），而根据第(3)步，输出结果的元素个数不超过 $|U|$ ，故算法第(2)步执行的次数不超过 $|U| - |X|$ 。

● 算法 6.1 修改版本（求属性集 X 关于函数依赖集 F 的闭包）

Input: X, F

Output: X_F^+

方法：计算 $X^{(i)}$ ($i = 1, 2, \dots, n$)

(1) $X^{(0)} := X, i = 0$

(2) 在 F 中寻找左边是 $X^{(i)}$ 的子集的函数依赖 $Y_j \rightarrow Z_j$ ($j = 1, 2, \dots, k$), $Y_j \subseteq X^{(i)}$,

在所有 Z_j 中寻找 $X^{(i)}$ 中未出现过的属性构成集合 A ;

若 A 为空集则转(4), 否则 $X^{(i+1)} = X^{(i)} \cup A$;

(3) 转 (2);

(4) 输出 $X^{(i)}$, 即为 X_F^+

算法6.1表述不同

例1: 已知关系模式 $R\langle U, F\rangle$, 其中

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

$$F=\{AB\rightarrow C, B\rightarrow D, C\rightarrow E, EC\rightarrow B, AC\rightarrow B\}。$$

求 $(AB)_F^+$ 。

解 设 $X^{(0)}=AB$;

(1)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:

$$AB\rightarrow C, B\rightarrow D。$$

于是 $X^{(1)}=AB\cup CD=ABCD。$

(2)因为 $X^{(0)} \neq X^{(1)}$ ，所以再找出左部为 $ABCD$ 子集的那些函数依赖，又得到 $AB \rightarrow C$, $B \rightarrow D$, $C \rightarrow E$, $AC \rightarrow B$,

于是 $X^{(2)} = X^{(1)} \cup BCDE = ABCDE$ 。

(3)因为 $X^{(2)} = U$ ，算法终止

所以 $(AB)_F^+ = ABCDE$ 。

例2: $R < U, F >$, $U = (A, B, C, G, H, I)$, $F = \{A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H\}$, 计算 $(AG)_F^+$

	所用依赖:	X^i
$X^0 = AG$	$A \rightarrow B$	AGB
	$A \rightarrow C$	AGBC
$X^1 = AGBC$	$CG \rightarrow H$	AGBCH
	$CG \rightarrow I$	AGBCH I
$(AG)_F^+ =$	ABCGHI	

例3

$R < U, F >, U = (A, B, C, D, E, G), F = \{A \rightarrow E, BE \rightarrow AG, CE \rightarrow A, G \rightarrow D\}$, 计算 $(AB)_F^+$

所用依赖

$A \rightarrow E$

$BE \rightarrow AG$

$G \rightarrow D$

$(AB)_F^+$

ABE

ABEG

ABEGD

再无函数
依赖可用



$(AB)_F^+$

= ABEGD

C属性有无
特殊之处?

综上所述：能够在足够少的时间内求出属性集 X 关于函数依赖集 F 的闭包，从而有效的判断某个函数依赖 $X \rightarrow Y$ 能否由Armstrong公理系统导出。

课外阅读内容：证明Armstrong公理系统的完备性
——每个 F 逻辑蕴涵的函数依赖都能够由Armstrong公理导出。
(不能导出则也不会被蕴含)

思路

回忆引理6.2：设 F 为属性集 U 上的一组函数依赖， $X、Y \subseteq U$ ，则 $X \rightarrow Y$ 能由 F 导出的充要条件是 $Y \subseteq X_F^+$ 。

X_F^+	$U - X_F^+$
1 1 1	0 0 0
1 1 1	1 1 1

该关系的特征：两个元组，一半重复，一半不重复

证明了逻辑蕴涵和根据Armstrong公理系统导出是等价的。



分解的正确性和函数依赖（集）密切相关，而合理的关系分解可利用Armstrong公理。

候选码的求解（基本方法）


对于给定的关系 $R(A_1, A_2, \dots, A_n)$ 和函数依赖集 F ，可将其属性分为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 的函数依赖左右两边均未出现的属性。

LR类：在 F 的函数依赖左右两边均出现的属性。

- 定理：对于给定的关系模式 $R\langle U, F \rangle$ ，若 X 是 L 类属性，则 X 必为 R 的**任何候选码的成员**。
- 推论：对于给定的关系模式 $R\langle U, F \rangle$ ，若 X 是 L 类属性，且 $X_F^+ = U$ ，则 X 必为 R 的**唯一候选码**。
- 定理：对于给定的关系模式 $R\langle U, F \rangle$ ，如果 X 是 R 类属性，则 X 必**不在 任何候选码中**。
- 定理：对于给定的关系模式 $R\langle U, F \rangle$ ，如果 X 是 R 的 N 类属性，则 X 必包括在 R 的**任何候选码的成员**。 
- 推论：对于给定的关系模式 $R\langle U, F \rangle$ ，如果 X 是 R 的 N 类和 L 类属性组成的集合，且 $X_F^+ = U$ ，则 X 必为 R 的**唯一候选码**。

问题：

已证明了Armstrong公理的有效性与完备性，因此，任意写出一个函数依赖集，可以很容易的推导出一个新的“等价”的函数依赖集。

使用哪一个？

为什么不用最小的那一个？

● 函数依赖集的等价性

定义：函数依赖集 F , G , 若 $F^+ = G^+$, 则称 F 与 G 等价, 或者说函数依赖集 F 与 G 互相覆盖。

引理6.3: $F^+ = G^+$ 充要条件是 $F \subseteq G^+$, $G \subseteq F^+$

证明: (充分性) 任选由 $X \rightarrow Y \in F$, 则由 $F \subseteq G^+$ 可得 X 的 F 闭包 X_F^+ 包含于 X 的 G^+ 闭包 $X_{G^+}^+$ 。

因此 $X \rightarrow Y \in (G^+)^+ = G^+$, 即 $F^+ \subseteq G^+$ 。

同理 $G^+ \subseteq F^+$ 。充分性得证。(必要性易知)

“闭包”是什么?
顾名思义。。。



判断两个函数依赖集等价的方法:

逐一判断 F 和 G 中的函数依赖 $X \rightarrow Y$ 是否能由对方导出。

等价的作用——

•最小依赖集/最小覆盖

定义：满足下列条件的函数依赖集 F 称为最小依赖集，亦称极小依赖集或者最小覆盖，记作 F_{\min} ：

- 单属性化**： F 中任一函数依赖 $X \rightarrow A$ ， A 必是单属性
- 无冗余化**： F 中不存在这样的函数依赖 $X \rightarrow A$ ，使得 F 与 $F - \{X \rightarrow A\}$ **等价**
- 既约化**： F 中不存在这样的函数依赖 $X \rightarrow A$ ，在 X 中有真子集 Z ，使得 F 与 $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ **等价**

右部

整个依赖关系

左部

[例2] 关系模式 $S\langle U, F\rangle$ ，其中：

$U=\{ \text{SNO}, \text{SDEPT}, \text{DPRES}, \text{CNAME}, \text{G} \}$,

{ 学号, 所在院系, 系主任, 课程名, 成绩}

$F=\{ \text{SNO}\rightarrow\text{SDEPT}, \text{SDEPT}\rightarrow\text{DPRES}, (\text{SNO}, \text{CNAME})\rightarrow\text{G} \}$

设 $F'=\{ \text{SNO}\rightarrow\text{SDEPT}, \text{SNO}\rightarrow\text{DPRES}, \text{SDEPT}\rightarrow\text{DPRES},$

$(\text{SNO}, \text{CNAME})\rightarrow\text{G}, (\text{SNO}, \text{SDEPT})\rightarrow\text{SDEPT} \}$

F 是最小覆盖，而 F' 不是。

因为： $F' - \{ \text{SNO}\rightarrow\text{DPRES} \}$ 与 F' 等价

$F' - \{ (\text{SNO}, \text{SDEPT})\rightarrow\text{SDEPT} \}$ 也与 F' 等价

最小依赖集在哪？

●定理6.3

每一个函数依赖集 F 都与一个极小函数依赖集 F_m 等价，此 F_m 称为 F 的最小依赖集。

证明思路：构造性的证明，找到一个求解等价最小依赖集的算法即可。



F 的最小依赖集的出现使关系的分解变得“明朗”、易于验证。

● 算法1——求解函数依赖集F的最小覆盖 F_m

①单属性化：逐个检查F中各函数依赖 $FD_i : X \rightarrow Y$ ，
若 $Y = A_1 A_2 \dots A_k$ ， $k \geq 2$ ，则用诸 $X \rightarrow A_i$ 代替Y

②无冗余化：逐个检查F中各函数依赖 $X \rightarrow A$ ，
令 $G = F - \{X \rightarrow A\}$ ，若 $A \in X_G^+$ ，则从F中去掉该函数依赖
(F与G等价 $\Leftrightarrow A \in X_G^+$)

X, G

③既约化：逐个检查“F”中各函数依赖 $X \rightarrow A$ ，
设 $X = B_1 \dots B_m$ ，逐个考查约减 B_i ($i=1, 2, \dots, m$)，
若 $A \in (X - B_i)_F^+$ ，则以 $(X - B_i)$ 取代X

X', F

(F与 $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$ 等价 $\Leftrightarrow A \in Z_F^+$ ，其中 $Z = X - B_i$)

例1: $F = \{A \rightarrow B, B \rightarrow A, A \rightarrow C, B \rightarrow C\}$, 求 F_{\min}

解: 检查 $A \rightarrow B$, $G = F - \{A \rightarrow B\} = \{B \rightarrow A, A \rightarrow C, B \rightarrow C\}$

$$(A)_G^+ = \{A, C\}, B \notin \{A, C\}$$

检查 $A \rightarrow C$, $G = F - \{A \rightarrow C\} = \{A \rightarrow B, B \rightarrow A, B \rightarrow C\}$

$$(A)_G^+ = \{A, B, C\}, C \in \{A, B, C\}$$

所以从 F 中删除 $A \rightarrow C$,

$$F_{\min} = \{A \rightarrow B, B \rightarrow A, B \rightarrow C\}$$

或者选择对函数依赖不同的检查顺序, 可得:

$$F_{\min} = \{A \rightarrow B, B \rightarrow A, A \rightarrow C\}$$

例2: $F = \{C \rightarrow A, A \rightarrow G, CG \rightarrow B, B \rightarrow A\}$, 求 F_{\min}

解: F 满足条件 (1)。

易知 F 可能违反条件 (2) 的只有 $C \rightarrow A$ 和 $B \rightarrow A$, 而
 $A \notin (C)_{F - \{C \rightarrow A\}}^+ = C$, $A \notin (B)_{F - \{B \rightarrow A\}}^+ = B$, 故 F 满足条件 (2)。

条件 (3): 判断 $CG \rightarrow B$,

$$(CG - C)_F^+ = (G)_F^+ = \{G\}$$

$$B \notin (CG - C)_F^+$$

$$(CG - G)_F^+ = (C)_F^+ = \{C, A, G, B\}$$

$$B \in (CG - G)_F^+, \text{ 以 } C \text{ 代替 } CG$$

$$\text{所以 } F_{\min} = \{C \rightarrow A, A \rightarrow G, C \rightarrow B, B \rightarrow A\}$$

???



■ F 的最小依赖集 F_m 不一定是唯一的, 它与对各函数依赖 FD_i 及 $X \rightarrow A$ 中 X 各属性的处置顺序有关。

算法存疑

算法1并不能保证结果为最小依赖集，举例分析过程如下：

假设经历了①②后剩余函数依赖 $X \rightarrow Y$ 、 $YW \rightarrow Z$ 和 $X \rightarrow Z$ ，可知该依赖集无②能去掉的函数依赖。

但是，不能排除之后经历③的处理上述函数依赖化简为 $X \rightarrow Y$ 、 $Y \rightarrow Z$ 和 $X \rightarrow Z$ 。此时则又产生了需要由②删除的函数依赖 $X \rightarrow Z$ 。

因此，按照①②③顺序的求解过程无法保证结果的最小化。

修改策略1

将算法改成①②③②③②③②③。。。的循环，循环终止条件为函数依赖集字符数（字符多次出现则重复计算）不再减少。

策略1正确性证明：假如②③在中途某次执行后函数依赖集和执行前没有变化，则算法再次迭代执行②③时输入条件不变、处理过程不变的情况下，输出亦不会有变化，因此算法若在继续迭代下去也无法找出新的违反函数依赖集最小化原则的情况，算法可正确终止。

假如②③每一次迭代都发现了新的违反函数依赖集最小化原则的情况，则每一次化简将导致函数依赖集的字符数减少至少一个，按照①②③的过程，不可能导致F变为空集，因此，算法会在某一次迭代后字符不再减少，亦即走到上一种情况的终止状态。从而算法可正确终止。

修改策略2

将算法改成①③②。

策略2正确性证明：反证法。

假设经过①③②后还可能化简，则只可能是出现了违反步骤③条件的情况。不妨描述为 $X \rightarrow Y$ 可以化简为 $X' \rightarrow Y$ （其中 X' 是 X 的真子集）。也就是说 X' 在经过②整体删除某一个或者多个冗余的函数依赖后的属性闭包是包含了 Y 的。

则按照属性闭包的定义， X' 在执行②之前的属性闭包也必然包含 Y ，但是这种情况在③执行后不可能出现，推出矛盾。故策略2是正确的。

易知：

- 一个最小函数依赖集去掉其中某个函数依赖后仍然是一个最小函数依赖集，但是二者不等价。
- 给定最小函数依赖集 $F = \{X \rightarrow A_m, \dots\}$,
 $G = F - \{X \rightarrow A_m\}$, 若对于可由 X 导出的属性集 Y , Y 的 F 闭包包含 A_m , 而 Y 的 G 闭包不包含 A_m , 则 $Y \rightarrow X$ 被 F 逻辑蕴含。

应用：算法6.3的证明之中。

6.4 模式的分解

●定义5.17:

- 函数依赖集合 $F_i = \{X \rightarrow Y \mid X \rightarrow Y \in F^+ \wedge X, Y \subseteq U_i\}$ 称为 F 在 U_i 上的投影

●定义5.16:

- 关系模式 $R \langle U, F \rangle$ 的一个分解是指

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

其中 $U = \bigcup_{i=1}^n U_i$ ，并且没有 $U_i \subseteq U_j$ ， $1 \leq i, j \leq n$ ， F_i 是 F 在 U_i 上的投影

6.4.1 分解正确性

分解不唯一

例R:

Sno	Sdept	Sloc
9901	计算机	D1
9902	电信	D2
9903	自控	D3
9904	电信	D2
9905	管理	D2

KEY: Sno

F: sno→sdept

sno→sloc

sdept→sloc

∴ R ∉ 3NF

1、分解1:

R1

Sno	Sdept
9901	计算机
9902	电信
9903	自控
9904	电信
9905	管理

R2

Sno	Sloc
9901	D1
9902	D2
9903	D3
9904	D2
9905	D2

[转至无损连接性](#)

2、分解2:

R3

Sno	Sloc
9901	D1
9902	D2
9903	D3
9904	D2
9905	D2

R4

Sdept	Sloc
计算机	D1
电信	D2
自控	D3
管理	D2

3、分解3:

R5

Sno	Sdept
9901	计算机
9902	电信
9903	自控
9904	电信
9905	管理

R6

Sdept	Sloc
计算机	D1
电信	D2
自控	D3
管理	D2

转至无损连接性

4、分解4: R7 (sno)、R8 (sdept)、R9 (sloc)
哪种分解正确的呢?

6.4.2 分解正确性标准

6.4.2.1、分解具有无损连接性

定义: 任给R (U、F), $\rho = \{R_1, R_2, \dots, R_n\}$ 是R的一个分解, 若对R的任一满足F的关系r都有:

$$r = \pi_{R_1}(r) \bowtie \pi_{R_2}(r) \bowtie \dots \bowtie \pi_{R_n}(r)$$

则称 ρ 是R满足F的一个**无损连接分解**。

其中:

π_{R_n} : 投影

\bowtie : 自然连接运算

可通过自然连接运算还原。

① 分解1

转至分解1

$$R1 \bowtie R2 = R$$

无损连接分解，不丢失信息（实际是指连接后不多出元祖，多出元祖即丢失原来真实元组集合描述的事实）。

② 分解2

转至分解2

$$R3 \bowtie R4 \neq R$$

多出三个元组：

(9902, D2, 管理)，

(9904, D2, 管理)，

(9905, D2, 电信)，

不具有无损连接性。

Sno	Sdept	Sloc
9901	计算机	D1
9902	电信	D2
9903	自控	D3
9904	电信	D2
9905	管理	D2
9902	管理	D2
9904	管理	D2
9905	电信	D2

③ 分解3

转至分解3

$$R5 \bowtie R6 = R$$

具有无损连接性。

④ 分解4

显然不能还原，丢失很多信息。

6.2.2.2、保持函数依赖性质

定义：任给 $R(U, F)$ ， $\rho = \{R_1, R_2, \dots, R_n\}$ 是 R 的一个分解，若 $F \Leftrightarrow \pi_{R_1}(F_1) \cup \pi_{R_2}(F_2) \cup \dots \cup \pi_{R_n}(F_n)$ ，则称 ρ 具有函数依赖保持性。

① 分解1

具有无损性连接性

未保持fd（丢失了fd: $sdept \rightarrow sloc$ ）

② 分解2

不具有无损连接性（多出两个元组）

未保持fd（丢失了fd: $sno \rightarrow sdept$ ）

③ 分解3

具有无损连接性

保持了fd（ $sno \rightarrow sdept, sdept \rightarrow sloc$, 但 $sdept \not\rightarrow sno$ ）

即 $sno \xrightarrow{t} sloc$

④ 分解4

不具有无损连接性
未保持fd

分解默认有损?



分解与连接的基本性质：引理6.4

分解-连接操作的
幂等?

关于正确分解的结论

- 1、若仅要求分解具有无损连接性，则分解一定能达到4NF。
- 2、若仅要求分解保持fd，则分解一定能达到3NF，但不一定能达到BCNF。
- 3、若既要求分解具有无损连接性，又保持了fd，则分解一定能达到3NF，但不一定能达到BCNF。

无损连接分解检验算法描述

（对原关系**n**列，分解为**k**个子关系的情况，构造**n**列**k**行的表**S**），例如：

A1	A2	A3	...	An
a_1	a_2	b_{13}	...	b_{1n}
b_{21}	a_2	a_3	...	b_{2n}
b_{31}	a_2	b_{33}	...	a_n

一行：一个子关系 一列：关系的属性

单元格：符号**a**表示所在列对应的属性包含于所在行对应的子关系，**b**则表示不包含。

符号**a**具有单下标，描述属性序号，符号**b**具有双下标，描述（子关系序号、属性序号）。

无损连接分解检验算法描述（先构造n列k行的表S）：

1.对F中每一个函数依赖 $X \rightarrow Y$ ，若S中存在元组 t_1 ， t_2 ，使得 $t_1[X] = t_2[X]$ ， $t_1[Y] \neq t_2[Y]$ ，则对每一个 $A_i \in Y$ ：

①若 $t_1[A_i]$ ， $t_2[A_i]$ 中有一个等于 a_j ，则另一个也改为 a_j ；

变a

②若①不成立，则取 $t_1[A_i] = t_2[A_i]$ （ t_2 的行号小于 t_1 ，即统一取行下标最小值）。

变b

2.反复执行1，直至：

①S中出现一行为 a_1, a_2, \dots, a_n 。

② S不再发生变化，且没有一行为 a_1, \dots, a_n 。

在①情况下， ρ 为无损分解，否则为有损分解。

每执行一次步骤1，表中的符号至少减少一个，表中符号有限，故算法执行能够在有效时间内完成。



定理6.4: ρ 为无损分解的充要条件是 S 中出现一行为 a_1, a_2, \dots, a_n 。

例1: $U = \{A, B, C, D, E\}$, $F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow E\}$
 $\rho = \{(A, B, C), (C, D), (D, E)\}$

$AB \rightarrow C$ 表格无变化

	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

	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

$C \rightarrow D$

	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	b_{25}
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

■ $F = \{AB \rightarrow C, C \rightarrow D, D \rightarrow E\}$

	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	b_{25}
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

$D \rightarrow E$

	A	B	C	D	E
ABC	a_1	a_2	a_3	a_4	a_5
CD	b_{21}	b_{22}	a_3	a_4	a_5
DE	b_{31}	b_{32}	b_{33}	a_4	a_5

S中存在一行全为“a”类符号， ρ 具有无损连接性。

无损连接分解 例2:

- $U = \{A, B, C, D, E\}$,
 $F = \{A \rightarrow B, B \rightarrow C, C \rightarrow B\}$
 $\rho = \{(A, C), (A, D), (B, C)\}$

$A \rightarrow B$

	A	B	C	D
AC	a_1	b_{12}	a_3	B_{14}
AD	a_1	b_{22}	b_{23}	a_4
BC	b_{31}	a_2	a_3	b_{34}

	A	B	C	D
AC	a_1	b_{12}	a_3	B_{14}
AD	a_1	b_{12}	b_{23}	a_4
BC	b_{31}	a_2	a_3	b_{34}

$$F = \{A \rightarrow B, B \rightarrow C, C \rightarrow B\}$$

 $B \rightarrow C$

	A	B	C	D
AC	a_1	b_{12}	a_3	B_{14}
AD	a_1	b_{12}	a_3	a_4
BC	b_{31}	a_2	a_3	b_{34}

 $C \rightarrow B$

	A	B	C	D
AC	a_1	a_2	a_3	B_{14}
AD	a_1	a_2	a_3	a_4
BC	b_{31}	a_2	a_3	b_{34}

- 分解的几个基本性质——引理6.4

- 定理6.5

• $R\langle U, F \rangle$ 的一个分解 $\rho = \{R_1\langle U_1, F_1 \rangle, R_2\langle U_2, F_2 \rangle\}$ 具有无损连接性的充要条件是：

$$U_1 \cap U_2 \rightarrow U_1 - U_2 \in F^+$$

或 $U_1 \cap U_2 \rightarrow U_2 - U_1 \in F^+$

每次无损的一
分为二

6.4.3 分解的算法

F的最小依赖集的出现使关系的分解变得“明朗”、易于验证。

●若仅要求分解保持函数依赖，则分解一定能达到3NF，但不一定能达到BCNF。

算法6.3

——转换为3NF的保持函数依赖的分解。

- 1) 将F“最小化”，得到 F_{\min} ；
- 2) 将 F_{\min} 中不出现的属性集划分为一个子关系；
- 3) 若 F_{\min} 中有函数依赖涉及到剩余的全部属性，则分解终止，否则转步骤4)；
- 4) 对 F_{\min} 按照相同左部分组，每组涉及到的属性集构成一个子关系，去掉其中可以被包含的子关系。

理解：

1) F 最小化之后按照相同的左部分组，消除了子关系中可能出现的非主属性对码的部分或传递函数依赖。

证明采用反证法，传递依赖的出现违背了最小依赖集的定义（若有传递，则有 $X \rightarrow Y$ ，且 $Y \not\rightarrow X$ ， $Y \rightarrow A_m$ 。令 $G = F - \{X \rightarrow A_m\}$ 则可推出 $X \rightarrow A_m$ （ $Y \rightarrow A_m$ ）依然逻辑蕴含于 G ，得出 F 并非最小依赖集的矛盾结论）。
否则 $Y \rightarrow X$

2) 该算法显然能够在有效时间内完成（最小依赖集有限）。

- 若既要求分解具有无损连接性，又保持了函数依赖，则分解一定能达到3NF，但不一定能达到BCNF。

算法6.4

——转换为3NF的保持函数依赖且无损连接的分解。

- 1) 按照算法6.3得到保持函数依赖的到3NF的分解；
- 2) 增加一个仅包含原关系主码的子关系（如果该子关系被算法6.3的某个子关系包含，则可以不增加该子关系）。

理解：

- 1) 满足3NF且保持函数依赖根据算法6.3即达到要求，**增加的子关系根据定理6.4可知，使分解具有无损连接性。**
- 2) 最小依赖集不唯一，因此分解的结果不唯一。
- 3) 同算法6.3，其执行能在有效时间内完成。

定理6.4: ρ 为无损分解的充要条件是 S 中出现一行为 a_1, a_2, \dots, a_n 。

有一个子关系包含码

●引理6.5

理解：1) 只要分解的每一步都是无损连接的，则整个分解就是无损连接的。

2) 一个无损连接的分解再增加新的子关系，所得的分解仍然是无损连接的。

●引理6.6 $(R1 \bowtie R2) \bowtie R3 = R1 \bowtie (R2 \bowtie R3)$

理解：即关系代数等价变换规则中连接的结合律。根据连接的定义证明。

(算法6.5、算法6.6作为课外阅读内容)

数据依赖的有效且完备的公理系统及其推理规则。

公理系统——

A1:函数依赖的自反律

若 $Y \subseteq X \subseteq U$, 则 $X \rightarrow Y$;

A2:函数依赖的增广律

若 $X \rightarrow Y$, $Z \subseteq U$, 则 $XZ \rightarrow YZ$;

A3:函数依赖的传递律

若 $X \rightarrow Y$, $Y \rightarrow Z$, 则 $X \rightarrow Z$;

A4:若 $X \twoheadrightarrow Y$, $V \subseteq W \subseteq U$, 则 $XW \twoheadrightarrow YV$;

A5:多值依赖的对称性

若 $X \twoheadrightarrow Y$, 则 $X \twoheadrightarrow Z$, 其中 $Z = U - X - Y$;

A6:多值依赖的传递性

若 $X \twoheadrightarrow Y$, $Y \twoheadrightarrow Z$, 则 $X \twoheadrightarrow Z - Y$;

A7:多值依赖的复制规则

若 $X \rightarrow Y$, 则 $X \twoheadrightarrow Y$;

A8:若 $X \twoheadrightarrow Y$, $W \rightarrow Z$, $W \cap Y = \emptyset$, $Z \subseteq Y$, 则 $X \rightarrow Z$;

推理规则:

1) 并规则

若 $X \rightarrow\!\!\rightarrow Y$, $X \rightarrow\!\!\rightarrow Z$, 则 $X \rightarrow\!\!\rightarrow YZ$;

2) 伪传递规则

若 $X \rightarrow\!\!\rightarrow Y$, $WY \rightarrow Z$, 则 $WX \rightarrow\!\!\rightarrow Z-WY$;

3) 混合伪传递规则

若 $X \rightarrow\!\!\rightarrow Y$, $XY \rightarrow Z$, 则 $X \rightarrow Z-Y$

4) 分解规则

若 $X \rightarrow\!\!\rightarrow Y$, $X \rightarrow Z$, 则 $X \rightarrow\!\!\rightarrow Y \cap Z$, $X \rightarrow\!\!\rightarrow Y-Z$,
 $X \rightarrow\!\!\rightarrow Z-Y$,

知识?

安全?

慕课讨论题

- 错误理解应用的语义规则对于模式设计有哪些影响？

如果未能正确理解应用的某些语义规则，可能会对关系模式设计造成什么影响？

- 如何设计一个关系规范化处理的程序？

如何设计一个程序，使其能对给定的关系模式自动地进行必要的关系规范化处理（比如规范到**3NF**）？