

课程名称: 数据库系统

关系模式设计优化

单位: 重庆大学计算机学院

- 如果关系模式存在冗余，你将用什么方法优化该模式？

主要目标

- 掌握优化关系模式的方法及规范
- 掌握函数依赖集闭包和属性集闭包
- 掌握无损分解和保持依赖的概念

思考问题

- 分解关系模式，是否就一定能达到优化的目的？分解关系模式可能存在什么问题？如何解决？

1.函数依赖和键（码）

- 给定 $R(A, B, C)$.
- $A \rightarrow ABC$ 意味着 A 是一个键（码）.
- 通常,
- $X \rightarrow R$ 意味着 X 是一个超键.
- 键的约束

$ssn \rightarrow did$

2.函数依赖集的闭包 F^+

- 函数依赖集的闭包？由 F 逻辑蕴含的所有函数依赖的集合。
- 计算函数依赖集的闭包：

```
 $F^+ = F$   
repeat  
  for each  $F^+$  中的函数依赖  $f$   
    在  $f$  上应用自反律和增补律  
    将结果加入到  $F^+$  中  
  for each  $F^+$  中的一对函数依赖  $f_1$  和  $f_2$   
    if  $f_1$  和  $f_2$  可以使用传递律结合起来  
      将结果加入到  $F^+$  中  
until  $F^+$  不再发生变化
```

F+例子

- $F = \{A \rightarrow B, B \rightarrow C, C D \rightarrow E\}$
- Step 1: F中的每一个函数依赖, 使用自反律
 - 得到: $CD \rightarrow C; CD \rightarrow D$
 - 加到F上:
 $F = \{A \rightarrow B, B \rightarrow C, C D \rightarrow E; CD \rightarrow C; CD \rightarrow D\}$
- Step 2: F中的每一个函数依赖, 使用增补律
 - $A \rightarrow B$ 得到: $A \rightarrow AB; AB \rightarrow B; AC \rightarrow BC; AD \rightarrow BD; ABC \rightarrow BC; ABD \rightarrow BD; ACD \rightarrow BCD$
 - $B \rightarrow C$ 得到: $AB \rightarrow AC; BC \rightarrow C; BD \rightarrow CD; ABC \rightarrow AC; ABD \rightarrow ACD, \text{etc.}$
- Step 3: 使用传递率
- 重复1~3步骤...

可以看出计算F+代价太高.

3. 属性集闭包

- (函数依赖集闭包的大小是 (属性的) 指数级的)
- 很多时候, 我们仅仅是想判断一个 FD $X \rightarrow Y$ 是否在 F 的闭包中. 一个有效的方式是:
 - 计算属性 X 的闭包 (记为 X^+):
 - X 的闭包 就是由 X 在 F 上蕴含的所有属性的集合。
 - 计算属性的闭包仅仅需要一个线性的时间算法就够了。
 - $F = \{A \rightarrow B, B \rightarrow C, C D \rightarrow E\}$ $A \rightarrow E$ 成立吗?

属性集闭包的计算

```
result :=  $\alpha$ ;  
repeat  
  for each 函数依赖  $\beta \twoheadrightarrow r$  in  $F$  do  
    begin  
      if  $\beta \subseteq \text{result}$  then  $\text{result} := \text{result} \cup \gamma$ ;  
    end  
until ( $\text{result}$  不变)
```

属性集闭包例子

- $F = \{A \rightarrow B, B \rightarrow C, CD \rightarrow E\}$

$A \rightarrow E$ 是否成立?

– 也就是, 判断 $A \rightarrow E$ 是否在 F^+ 中?

等价于, E 是否在 A^+ 中?

- Step 1: $\text{Result} = A$
- Step 2: 考虑 $A \rightarrow B$, $\text{Result} = AB$

考虑 $B \rightarrow C$, $\text{Result} = ABC$

考虑 $CD \rightarrow E$, CD 不在 ABC , 不添加

- Step 3: $A^+ = \{ABC\}$

属性集闭包例子

- $F = \{A \rightarrow B, AC \rightarrow D, AB \rightarrow C\}$?
- 计算 A^+ 。
- Answer: $A^+ = ABCD$

属性集闭包例子

- $R = (A, B, C, G, H, I)$
 $F = \{A \rightarrow B; A \rightarrow C; CG \rightarrow H; CG \rightarrow I; B \rightarrow H\}$
 $(AG)^+ = ?$
- Answer: ABCGHI
- AG 是候选键吗?
 - 这个问题包括两部分:
 1. AG 是一个超键吗?
 - $AG \rightarrow R? \iff Is (AG)^+ \supseteq R$
 2. AG 的子集是否是一个超键?
 - $A \rightarrow R? \iff Is (A)^+ \supseteq R$
 - $G \rightarrow R? \iff Is (G)^+ \supseteq R$

属性集闭包的作用

- 属性集闭包的作用:

1. 测试超键:

- 判断 X 是否是一个超键? 只需要计算 X^+ , 检查 X^+ 是否包括 R 的所有属性.

2. 检测函数依赖

- 判断 $X \rightarrow Y$ 是否成立 (或者说, 是否在 F^+ 中), 只需要判断 $Y \subseteq X^+$.
- 因此, 我们计算 X^+ , 然后检测这个属性集闭包是否包括 Y .
- 简单有用的方法

3. 计算 F 的函数依赖集闭包

计算 F^+

- $F = \{ A \rightarrow B, B \rightarrow C \}$. 计算 F^+ (属性包括 A, B, C).

Step 1: 构建一个空的二维表, 行和列列出所有可能的属性组合

	A	B	C	AB	AC	BC	ABC
A							
B							
C							
AB							
AC							
BC							
ABC							

Step 3: 将结果填写到二维表中

Step 2: 计算所有的属性组合的属性集闭包

Attribute closure
$A^+ = ?$
$B^+ = ?$
$C^+ = ?$
$AB^+ = ?$
$AC^+ = ?$
$BC^+ = ?$
$ABC^+ = ?$

计算 F^+

- $F = \{ A \rightarrow B, B \rightarrow C \}$. 计算 F^+ (属性包括 A, B, C).

- 例如: A^+ .

Step 1: $\text{Result} = A$

Step 2: 考虑 $A \rightarrow B$, $\text{Result} = A \cup B = AB$

考虑 $B \rightarrow C$, $\text{Result} = AB \cup C = ABC$

Step 3: $A^+ = \{ABC\}$

Computing F+

$F = \{ A \rightarrow B, B \rightarrow C \}$. 计算 F^+ (包括属性 A, B, C).

Step 1: 构建一个空的二维表, 行和列列出所有可能的属性组合

	A	B	C	AB	AC	BC	ABC
A	√	√	√	√	√	√	√
B							
C							
:							

Step 3: 将结果填写到二维表中.

由于 $A^+ = ABC$. 填写标的时候, 考虑第一列, A 是 A^+ 的一部分吗? 是, 勾选. B 是 A^+ 的一部分吗? 是, 勾选...

Step 2: 计算所有的属性组合的属性集闭包

Attribute closure
$A^+ = \mathbf{ABC}$
$B^+ = ?$
$C^+ = ?$
$AB^+ = ?$
$AC^+ = ?$
$BC^+ = ?$
$ABC^+ = ?$

计算F+

- $F = \{ A \rightarrow B, B \rightarrow C \}$. Compute F^+ (包括属性A, B, C).

	A	B	C	AB	AC	BC	ABC
A	✓	✓	✓	✓	✓	✓	✓
B		✓	✓			✓	
C			✓				
AB	✓	✓	✓	✓	✓	✓	✓
AC	✓	✓	✓	✓	✓	✓	✓
BC		✓	✓			✓	
ABC	✓	✓	✓	✓	✓	✓	✓

Attribute closure
$A^+ = ABC$
$B^+ = BC$
$C^+ = C$
$AB^+ = ABC$
$AC^+ = ABC$
$BC^+ = BC$
$ABC^+ = ABC$

- 每一个✓ 表示FD (行) \rightarrow (列) 在 F^+ 中.
- 每一个✓ (列) 在(行)+中

计算F+

- $F = \{ A \rightarrow B, B \rightarrow C \}$. Compute F^+ (包括属性A, B, C).

$A \rightarrow BC$

	A	B	C	AB	AC	BC	ABC
A	√	√	√	√	√	√	√
B		√	√			√	
C			√				
AB	√	√	√	√	√	√	√
AC	√	√	√	√	√	√	√
BC		√	√			√	
ABC	√	√	√	√	√	√	√

Attribute closure
$A^+ = ABC$
$B^+ = BC$
$C^+ = C$
$AB^+ = ABC$
$AC^+ = ABC$
$BC^+ = BC$
$ABC^+ = ABC$

- 每一个√ 表示FD (行) \rightarrow (列) 在 F^+ 中.
- 每一个√ (列) 在(行)+中

4 模式分解的基本标准

S	N	L	R	W	H
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Original relation
(not stored in DB!)

Decomposition
(in the DB)

=

S	N	L	R	H
123-22-3666	Attishoo	48	8	40
231-31-5368	Smiley	22	8	30
131-24-3650	Smethurst	35	5	30
434-26-3751	Guldu	35	5	32
612-67-4134	Madayan	35	8	40



R	W
8	10
5	7

分解的问题

- 模式分解可能存在三种问题:
 - (1) 一些查询可能会代价变高.
e.g., Attishoo 挣了多少钱? ($\text{earn} = W * H$)
 - (2) 分解后, 根据分解的实例, 我们可能不能重新构建分解前的实例!
 - (3) 检查某些依赖需要考虑分解后的多个关系.
- 折中: 考虑这些问题 vs. 冗余.

分解

- 将分解符合3NF或更高的范式是一种很好的保证方法.
- **所有分解应该是无损的! (*Avoids Problem (2)*)**

分解问题2

Student_ID	Name	Dcode	Cno	Grade
123-22-3666	Attishoo	INFS	501	A
231-31-5368	Guldu	CS	102	B
131-24-3650	Smethurst	INFS	614	B
434-26-3751	Guldu	INFS	614	A
434-26-3751	Guldu	INFS	612	C

≠

Name	Dcode	Cno	Grade
Attishoo	INFS	501	A
Guldu	CS	102	B
Smethurst	INFS	614	B
Guldu	INFS	614	A
Guldu	INFS	612	C

▷◁

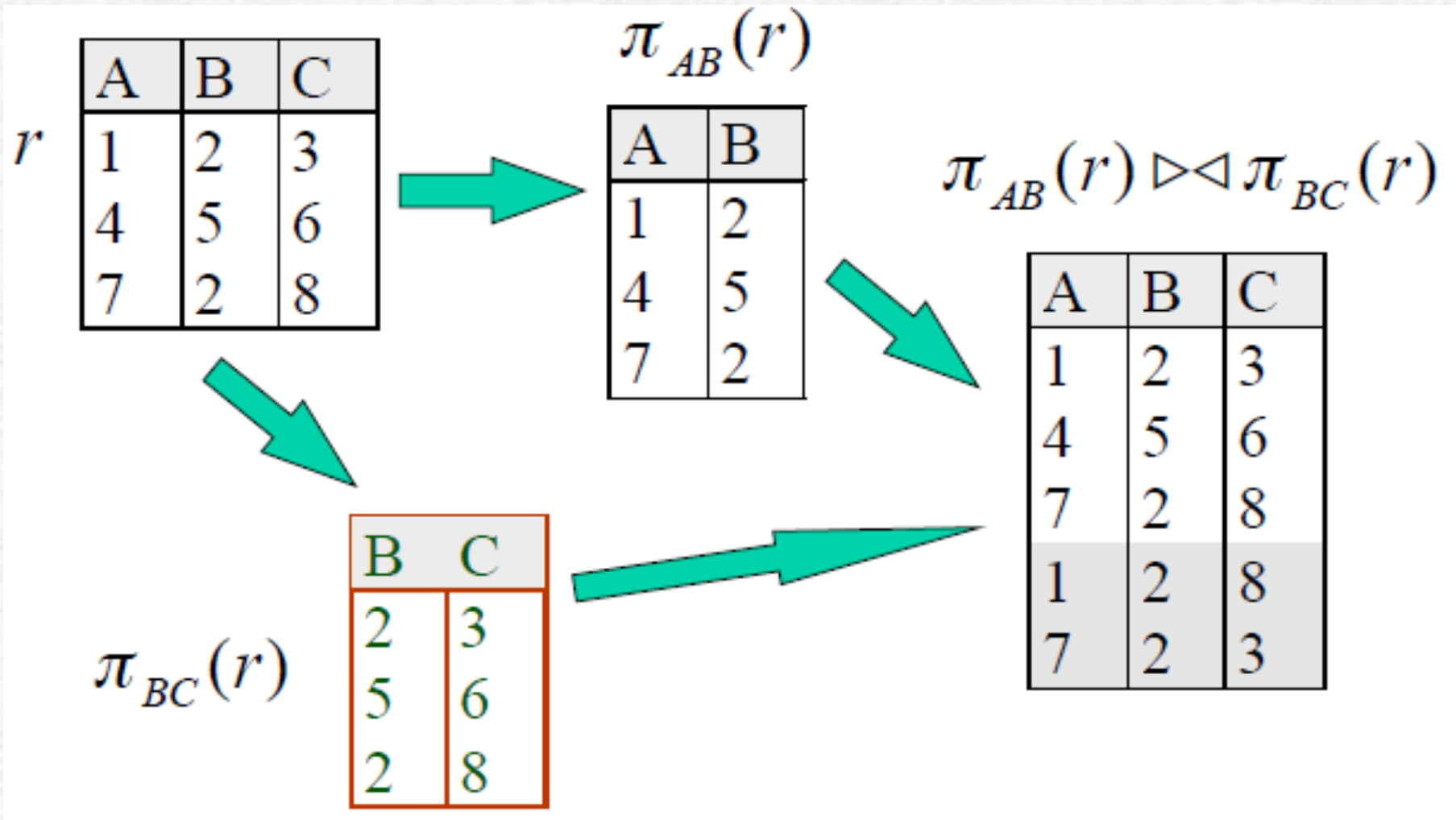
Student_ID	Name
123-22-3666	Attishoo
231-31-5368	Guldu
131-24-3650	Smethurst
434-26-3751	Guldu

无损连接分解

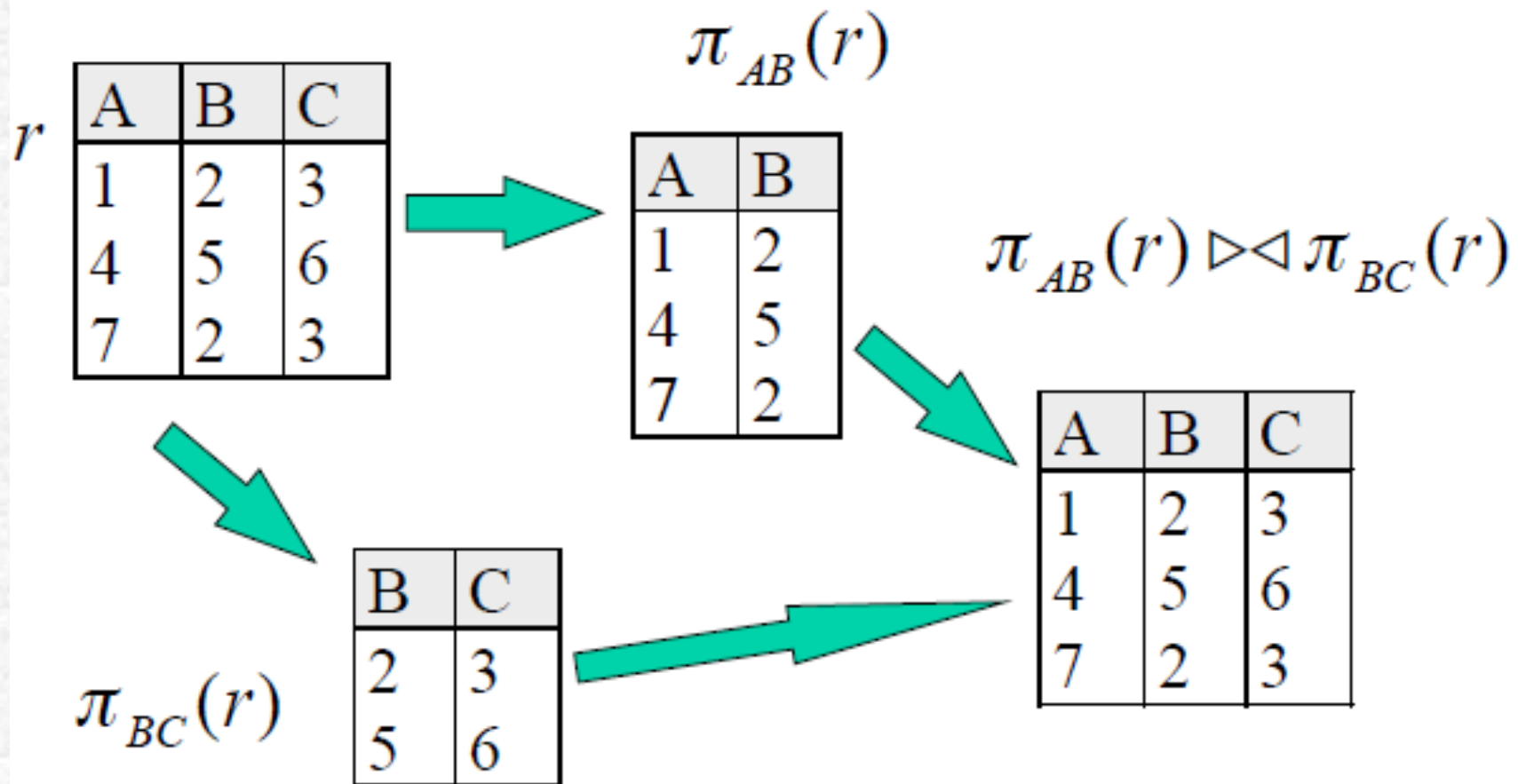
- 将R 分解为 R1 何R2 ，如果是无损连接分解，那么应该满足：

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

例子（不是无损的）



例子 (无损的)



We have $(AB \cap BC) \rightarrow BC$

无损连接分解

- 将 R 分解为 R_1 何 R_2 是无损分解，如果下面至少一个成立的话，那么分解是无损分解：

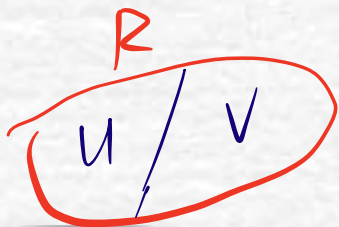
(仅适用于分解为两模式情形)

- $R_1 \cap R_2 \rightarrow R_1$ (函数依赖)
- $R_1 \cap R_2 \rightarrow R_2$ (函数依赖)


交集是 R_1 的超码

实际上将 R 分解为 (UV) 和 $(R-V)$ ，如果 $U \rightarrow V$ 在 R 上成立，那么分解是无损连接分解

$$UV \cap R - V = U$$



保持函数依赖

- $R = (A, B, C)$
 $F = \{A \rightarrow B, B \rightarrow C\}$
 - 分解方式可能如下:
 - $R1 = (A, B), R2 = (B, C)$ $B \rightarrow BC$
 - 无损连接分解:
 $R1 \cap R2 = \{B\}$ and $B \rightarrow BC$
 - 保持函数依赖
 - $R1 = (A, B), R2 = (A, C)$ $A \rightarrow AB$
 - 无损连接分解:
 $R1 \cap R2 = \{A\}$ and $A \rightarrow AB$
 - 没有保持函数依赖
(不能检测 $B \rightarrow C$ $R1 \not\bowtie R2$) 

保持函数依赖

- 保持函数依赖的分解（直观上）：
 - R 分解为X, Y 和Z, 函数依赖集FDs在X, Y, Z上成立, 那么FDs也会在R上成立。
(Avoids Problem (3))

分解后的函数依赖?

- 函数依赖集的投影: R 分解为 X, \dots F 在 X 上的投影 (denoted F_X) 是如下的 FDs $U \rightarrow V$ in F^+ (*closure of F*), U, V 是 X 中的属性.

保持函数依赖的分解

- 将R 分解为X 和Y 是保持函数依赖的，当且仅当 $(F_X \cup F_Y)^+ = F^+$
 - i.e., if we consider only dependencies in the closure F^+ that can be checked in X without considering Y, and in Y without considering X, these imply all dependencies in F^+ .
- 注意是 F^+ ，而不是F。
- 保持函数依赖并不能保证保持无损连接分解。
- 反之亦然。

判断两个函数依赖集是否等价

- 如果 $F1^+ = F2^+$, 那么 $F1$ 和 $F2$ 等价.
- 例如, $F1 = \{A \rightarrow B, A \rightarrow C\}$ 和 $F2 = \{A \rightarrow BC\}$ 等价。
- 怎么测试? Two steps:
 - *Every FD in $F1$ is in $F2^+$*
 - *Every FD in $F2$ is in $F1^+$*
- 这两步都需要多次使用属性集的闭包 (many times) for X^+

保持函数依赖

- $R = (A, B, C)$
 $F = \{A \rightarrow B, B \rightarrow C\}$
 - 分解方式可能如下:
- $R1 = (A, B), R2 = (B, C)$

– 无损连接分解:

$$R1 \cap R2 = \{B\} \text{ and } B \rightarrow BC$$

– 保持函数依赖

- $R1 = (A, B), R2 = (A, C)$

– 无损连接分解:

$$R1 \cap R2 = \{A\} \text{ and } A \rightarrow AB$$

– 没有保持函数依赖

(不能检测 $B \rightarrow C$ $R1$ $R2$)

$$F_{R1}: A \rightarrow B$$

$$F_{R2}: B \rightarrow C$$

$$\Rightarrow F: \begin{matrix} A \rightarrow B \\ B \rightarrow C \end{matrix}$$

$$F_{R1}: A \rightarrow B$$

$$F_{R2}: A \rightarrow C$$

~~\Rightarrow~~ $F: \begin{matrix} A \rightarrow B \\ B \rightarrow C \end{matrix}$
No!



例子

- $F = \{ A \rightarrow BC, B \rightarrow C \}$. 判断 $C \rightarrow AB$ 是否在 F^+ ?

- Answer: 不在.

Reason 1) $C^+ = C$, 不包括 AB .

Reason 2) 反例, 不存在 $C \rightarrow AB$.

A	B	C
1	1	2
2	1	2

例子

- $R(A, B, C, D, E)$,
- $F = \{A \rightarrow B, C \rightarrow D\}$
- 候选键?
- ACE.
- 怎么计算?
- Intuitively,
 - A is not determined by any other attributes (like E), and A has to be in a candidate key (because a candidate key has to determine all the attributes).
 - Now if A is in a candidate key, B cannot be in the same candidate key, since we can drop B from the candidate without losing the property of being a “key”.
 - So B cannot be in a candidate key
 - Same reasoning apply to others attributes.

本讲小结

- 函数依赖和码
- 属性集闭包
- 模式分解的标准

