目录

数据库

- 一些基本定义
  - 关系
    - **关系**：描述实体、属性、实体间的联系。从形式上看，它是一张**二维表**
    - **关系模式**：相当于关系的**表头**
    - **元组**：指代关系这个二维表中的**行**
    - **关系实例**：指代某**一组特定的行**，这组行来自关系中
    - **属性**：指代**列**
    - **域**：对于关系中每个属性，都存在一个**允许取值的集合**，称为该属性的域；要求对每个属性的域都是原子的，即不可再分

      eg. instructor 表中的一个属性 phone_number，存放一组电话号码，可以被分为单个电话号码，所以该域不是原子的
  - 各种码
    - **超码**：**一个或多个**属性的集合，这些属性的**组合**可以在一个关系中**唯一的标识一个元组。**

      若 K 是一个超码，那么包含 k 的任意集合（超集）也是超码
    - **候选码**：**最小的超码**；也就是说候选码的任意真子集都不能成为超码

      eg. {ID,name}不能成为候选码，因为单独的{ID}已经是候选码
    - **主码**：设计者**选中的**用来在一个关系中区分不同元组的**候选码**
    - **外码**：其他关系的主码
    - **主属性**：包含在候选码的属性集中的属性
    - **非主属性**：除了主属性之外的属性
  - 范式
    - **函数依赖**（FDs）
      - 形式化表示：X->Y

        X 和 Y 是属性集
      - 理解：关系 r 中给定两个元组，如果两个元组的 X 属性值相等，那么 Y 属性值也必相等；也可以理解为函数的映射
      - **平凡**和**非平凡**的函数依赖

在关系模式R(U)中，对于U的子集X和Y，

如果X→Y，但Y ⊆ X，则称X→Y是非平凡的函数依赖

若X→Y，但Y ⊆ X，则称X→Y是平凡的函数依赖

例：在关系SC(Sno, Cno, Grade)中，

非平凡函数依赖：(Sno, Cno) → Grade

平凡函数依赖：(Sno, Cno) → Sno

(Sno, Cno) → Cno

其实就是，如果 Y 是 X 的子集，那么 X->Y 就是平凡的函数依赖

- **完全**函数依赖和**部分**函数依赖

定义8.2 在关系模式R(U)中，如果X→Y，并且对于X的任何一个真子集X'，都有

$X' \not\rightarrow Y$，则称Y完全函数依赖于X，记作$X \xrightarrow{f} Y$。

若X→Y，但Y不完全函数依赖于X，则称Y部分函数依赖于X，记作$X \xrightarrow{P} Y$。

例：在关系SC(Sno, Cno, Grade)中，

由于：$Sno \not\rightarrow Grade$，$Cno \not\rightarrow Grade$，

因此：$(Sno, Cno) \xrightarrow{f} Grade$

其实就是，如果 X 的某个真子集也能和 Y 产生函数依赖，那么 Y 就部分函数依赖于 X

- **传递**函数依赖

定义8.3 在关系模式R(U)中，如果X→Y，Y→Z，且$Y \not\subseteq X$，$Y \not\rightarrow X$，则称Z传递函数依赖于X。

注：如果Y→X，即X←→Y，则Z直接依赖于X。

例：在关系Std(Sno, Sdept, Mname)中，有：

Sno → Sdept，Sdept → Mname

Mname传递函数依赖于Sno

其实传递函数依赖注意一点，就是 X->Y，但是 Y-|>X。 比如例子中的学生 ID（Sno）->学生系（Sdept），但是 Sdept-|>Sno

- **几种范式**
  - **1NF**：每个属性都是原子属性；

    本质上所有关系都满足第一范式
  - **2NF**

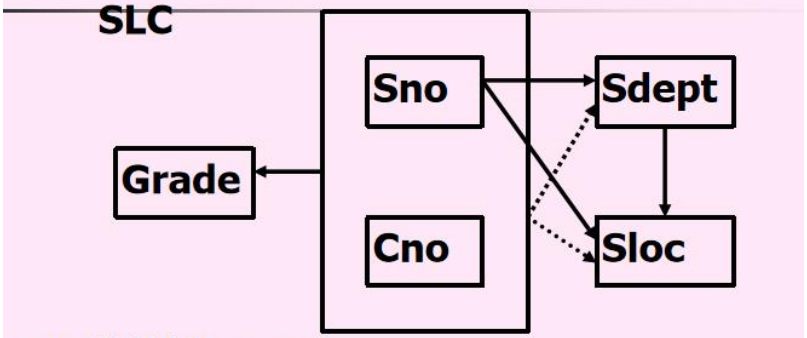
定义8.6 若关系模式R∈1NF，并且每一个非主属性都完全函数依赖于R的码，则R∈2NF。

- 补充
  - 分解说明
    - 采用投影分解法将一个1NF的关系分解为多个2NF的关系，可以在一定程度上减轻原1NF关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。
    - 将一个1NF关系分解为多个2NF的关系，并不能完全消除关系模式中的各种异常情况和数据冗余。
  - 举例

例: 关系模式 SLC(Sno, Sdept, Sloc, Cno, Grade)
Sloc为学生住处，假设每个系的学生住在同一个地方。

- 函数依赖包括：

$$(Sno, Cno) \xrightarrow{f} Grade$$
$$Sno \rightarrow Sdept$$
$$(Sno, Cno) \xrightarrow{P} Sdept$$
$$Sno \rightarrow Sloc$$
$$(Sno, Cno) \xrightarrow{P} Sloc$$
$$Sdept \rightarrow Sloc$$



- SLC的码为(Sno, Cno)
- SLC满足第一范式。
- 非主属性Sdept和Sloc部分函数依赖于码(Sno, Cno)
- 原因

  Sdept、 Sloc部分函数依赖于码。
- 解决方法

SLC分解为两个关系模式，以消除这些部分函数依赖

SC（Sno， Cno， Grade）
SL（Sno， Sdept， Sloc）

例：SLC(Sno, Sdept, Sloc, Cno, Grade) ∈1NF

SLC(Sno, Sdept, Sloc, Cno, Grade) ∉2NF

SC（Sno， Cno， Grade） ∈ 2NF

SL（Sno， Sdept， Sloc） ∈ 2NF

- **3NF**

  关系模式 R<U，F> 中不存在非主属性对于码的**传递函数依赖**（如 X→Y，Y→Z，X 是主键），则称 R<U，F> ∈ 3NF。

  - 补充
    - 若 R∈3NF，则 R 的每一个非主属性既不部分函数依赖于候选码也不传递函数依赖于候选码。
    - 分解说明

      采用投影分解法将一个2NF的关系分解为多个3NF的关系，可以在一定程度上解决原2NF关系中存在的插入异常、删除异常、数据冗余度大、修改复杂等问题。

      将一个2NF关系分解为多个3NF的关系后，并不能完全消除关系模式中的各种异常情况和数据冗余。

    - 举例

      例：2NF关系模式SL(Sno, Sdept, Sloc)中

      函数依赖：

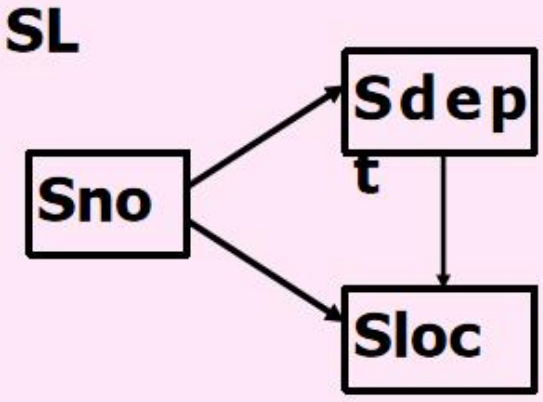
      Sno→Sdept

      Sdept→Sloc

      Sno→Sloc

      Sloc传递函数依赖于Sno，即SL中存在非主属性对码的传递函数依赖。

      函数依赖图：

- 解决方法

  采用投影分解法，把SL分解为两个关系模式，以消除传递函数依赖：

  SD（Sno， Sdept）

  DL（Sdept， Sloc）

  SD的码为Sno， DL的码为Sdept。

- **BCNF**
  - 对于 R 有函数依赖集 FDs F ， 如果 R 符合 BCNF 当且仅当每一个非平凡的 FD X → A in F , X 是 R 的超键。也就是说, **R 符合 BCNF 当且仅当非平凡的函数依赖箭头左侧是超键。如果 R 只有两个属性，那么它符合 BCNF。**
  - 关系 R，函数依赖集 F 中每一个函数依赖 X→A (X、A⊆ R)，符合下列描述中的一个:
    - – A ⊆ X (平凡的 FD), or
    - – X 是超键
  - 补充
    - BCNF 要求，对于每个非平凡的函数依赖，左侧必定为超键
    - 一些性质

      1. 所有非主属性都完全函数依赖于每个候选码
      2. 所有主属性都完全函数依赖于每个不包含它的候选码
      3. 没有任何属性完全函数依赖于非码的任何一组属性

    - 3NF 与 BCNF 关系

      3NF与BCNF的关系

      - 如果关系模式R∈BCNF，必定有R∈3NF
      - 如果R∈3NF，且R只有一个候选码，则R必属于BCNF。

    - 举例

      例：在关系模式STJ（S，T，J）中，S表示学生，T表示教师，J表示课程。

      - 每一教师只教一门课。每门课由若干教师教，某一学生选定某门课，就确定了一个固定的教师。某个学生选修某个教师的课就确定了所选课的名称 ：
      - (S，J)→T，(S，T)→J，T→J

STJ∈3NF，理由：

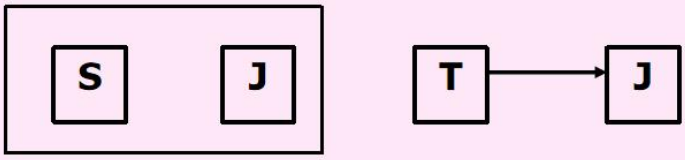
- (S，J)和(S，T)都可以作为候选码
- S、T、J都是主属性

STJ∉BCNF，理由：

- T→J，T是决定属性集，T不是候选码

解决方法：将STJ分解为二个关系模式：

SJ(S，J) ∈ BCNF，　TJ(T，J)∈ BCNF



ST　　TJ

没有任何属性对码的部分函数依赖和传递函数依赖

- 规范化

11. 规范化

- 定义：一个低一级范式的关系模式，通过模式分解可以转换为若干个高一级范式的关系模式集合，这种过程就叫关系模式的规范化。
- 步骤：　1NF
  ↓ 消除非主属性对码的部分函数依赖
  2NF
  ↓ 消除非主属性对码的传递函数依赖
  3NF
  ↓ 消除主属性对码的部分和传递函数依赖

BCNF

↓ 消除非平凡且非函数依赖的多值依赖

4NF

- **属性集闭包**
  - 算法

```
result := α;
    repeat
        for each 函数依赖 β →(γ) in F do
            begin
                if β ⊆ result then result := result ∪ γ;
            end
    until (result 不变)
```

  - 举例

属性集闭包例子

- F = {A → B, B → C, CD → E }
  A →E是否成立?
  - 也就是，判断 A → E 是否在 F+中?
    等价于，E 是否在A+中?
- Step 1: Result = A
- Step 2: 考虑A → B, Result = AB
  考虑B → C, Result = ABC
  考虑CD → E, CD 不在ABC, 不添加
- Step 3: A+ = {ABC}
  E 不在A+中,所以 A → E 不在F+中

- 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 → Y 是否成立（或者说，是否在*F+中)，只需要判断 Y ⊆ 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*

- 如何计算 F+

- F={ A → B, B → 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							

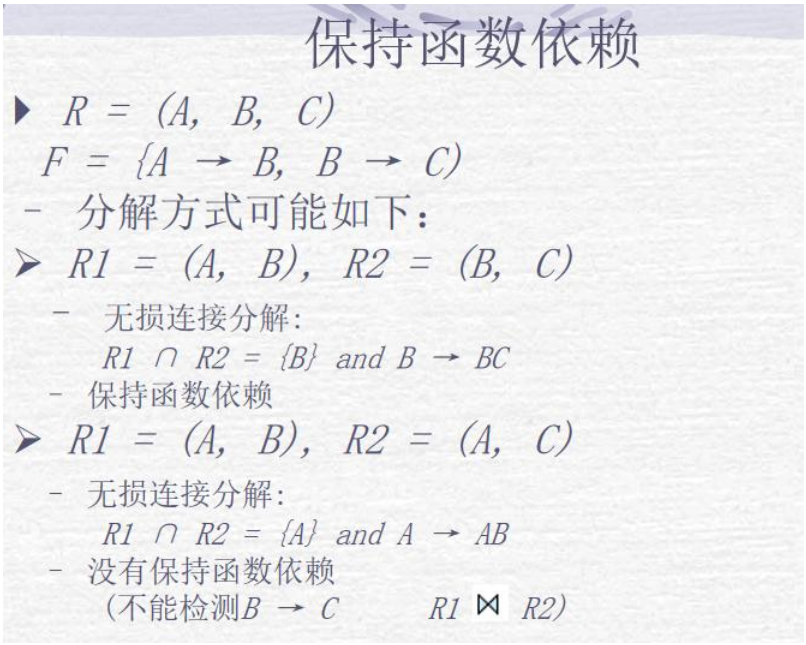
Step 2: 计算所有的属性组合的属性集闭包

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

Step 3: 将结果填写到二维表中

- **分解**
  - 无损连接分解

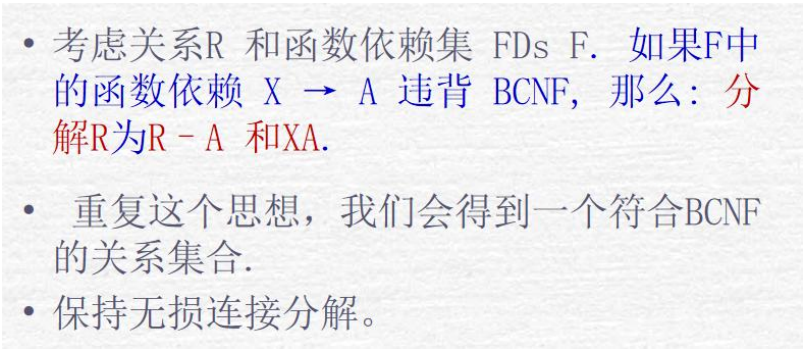
  - 保持函数依赖



- **BCNF 的检测与分解**
  - 检测

    关系 R，函数依赖集 F 中每一个函数依赖 X→A (X、A⊆ R)，符合下列描述中的一个:
    - – A ⊆ X (平凡的 FD), or
    - – X 是超键

    即判断每个函数依赖是否符合上述两个中的一个
    - 1.首先找出 R 的超键
    - 2.列出所有的非平凡函数依赖
    - 3.确认每一个函数依赖箭头左边的属性集是 R 的超键。
  - 分解
    - 仅考虑保持无损连接分解（缺点是分解后可能不会保持函数依赖）

    - 要想保持函数依赖，那么就定义一个弱的关系，即 3NF。3NF 一般保持无损连接分解和保持函数依赖的
- **3NF 的检测与分解**
  - 检测
    - 关系 R，函数依赖集 F 中每一个函数依赖 X→A (X、A ⊆ R)，符合下列描述中的一个:
      - – A ⊆ X (平凡的 FD), or
      - – X 是超键, or
      - – A 是 R 的候选键的一部分

即判断每个函数依赖是否符合上述三个中的一个

- 和 BCNF 相同，多增加判断第三个条件
- 即，若不是 1,2 条件，那么就要判断函数依赖右侧是否是主属性，若是，则满足条件；若不是，则非 3NF

- 分解：3NF 合成算法
  - 需要的概念
    - 无关属性

如果去除函数依赖中的一个属性不改变该函数依赖集的闭包，则称该属性是**无关的**(extraneous)。

无关属性(extraneous attribute)的形式化定义如下：考虑函数依赖集 F 及 F 中的函数依赖 $\alpha \rightarrow \beta$。

- 如果 $A \in \alpha$ 并且 F 逻辑蕴含 $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$，则属性 A 在 α 中是无关的。
- 如果 $A \in \beta$ 并且函数依赖集 $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ 逻辑蕴含 F，则属性 A 在 β 中是无关的。

    - 对于一个α->β：首先看α中的属性，如果去掉某个属性之后，

若要判断的属性位于依赖的**左侧**，例如{undefinedAB→C}，则删除该属性，在原本的依赖集F中计算该依赖左部集合的闭包α+。若α+闭包中包含该依赖右侧所有的属性，则该属性则是无关属性

例3：函数依赖集F{Z→X，X→P，XY→WP，XYP→ZW}

关注到XYP中的P属性。它在左侧。删除它，求解左侧剩下属性集XY的闭包，求取域是原来的F{Z→X，X→P，XY→WP，XYP→ZW}。因为XY→WP，所以（XY）+ = XYWP；又因为XYP→ZW，所以（XY）+ = XYWPZ，包含依赖右侧ZW

故P是无关属性

例4：F{A→BC，B→C，AB→C}

关注AB→C的A属性。删除它后余下的部分B→C在F中已经存在，故A是无关属性

    - 然后看β中的属性，如果去掉某个属性之后，还是能够通过其他函数依赖推导出

无关属性的判断（又称冗余🔍）：

给定一个函数依赖集F：

若要判断的属性位于依赖的**右侧**，例如{AB→C，……}，则删除该属性，在余下的函数依赖集F'中计算该依赖左部集合的闭包α+。若α+中包含要判断的属性，则该属性就是无关属性（冗余）

例2：F{A→BC，B→AC，C→AB}

关注B属性。它在依赖右侧。删除该属性，余下F'={A→C，B→AC，C→AB},计算左侧剩余属性集（A）的闭包（A）+。

因为A→C，C→AB，所以（A）+中包含删去的属性B

故B是无关属性

    - 正则覆盖

F 的**正则覆盖**(canonical cover) F_c 是一个依赖集，使得 F 逻辑蕴含 F_c 中的所有依赖，并且 F_c 逻辑蕴涵 F 中的所有依赖。此外，F_c 必须具有如下性质：

- F_c 中任何函数依赖都不含无关属性。
- F_c 中函数依赖的左半部都是唯一的。即，F_c 中不存在两个依赖 $\alpha_1 \rightarrow \beta_1$ 和 $\alpha_2 \rightarrow \beta_2$，满足 $\alpha_1 = \alpha_2$。

$F_c = F$

repeat

使用合并律将 F_c 中所有形如 $\alpha_1 \rightarrow \beta_1$ 和 $\alpha_1 \rightarrow \beta_2$ 的依赖替换为 $\alpha_1 \rightarrow \beta_1\beta_2$

在 F_c 中寻找一个函数依赖 $\alpha \rightarrow \beta$，它在 α 或在 β 中具有一个无关属性

/* 注意，使用 F_c 而非 F 检验无关属性 */

如果找到一个无关属性，则将它从 F_c 中的 $\alpha \rightarrow \beta$ 中删除

until（F_c 不变）

    - 定义

正则覆盖的求法与判断属性是否冗余

正则覆盖定义：一个依赖集F的正则覆盖Fc也是一个依赖集，F逻辑蕴含Fc中所有依赖，且Fc逻辑蕴涵F的所有依赖。Fc满足以下两个特性

·Fc中任何 函数依赖 都不含无关属性

·Fc中函数依赖左半部分都是唯一的，不允许重复

- 求解

  正则覆盖求解法：

  一.**使用合并律将所有左部相同的函数依赖合并成一个**

  例1：设有一函数依赖集F，其中有{A→B，A→E，……}，则将这两个函数依赖合并为{A→BE，……}

  二.**在合并后的函数依赖集中寻找一个无关属性，将它删除**

  无关属性的定义是什么呢？下面马上就提到

  三.重复1、2步骤，直到依赖集不再发生变化

- 举例

  一道完整的正则覆盖的题目：

  例5：F={Z→X，X→P，XY→WP，XYP→ZW}，求出一个正则覆盖

  第一次循环，未发现可合并项；直接快进至寻找无关属性

  首先关注到XYP→ZW的P属性。根据例题③的步骤，得知它是无关属性，故删除，F'={Z→X，X→P，XY→WP，XY→ZW}

  第二次循环。发现可以合并{XY→WP，XY→ZW}，得F'={Z→X，X→P，XY→PWZ}

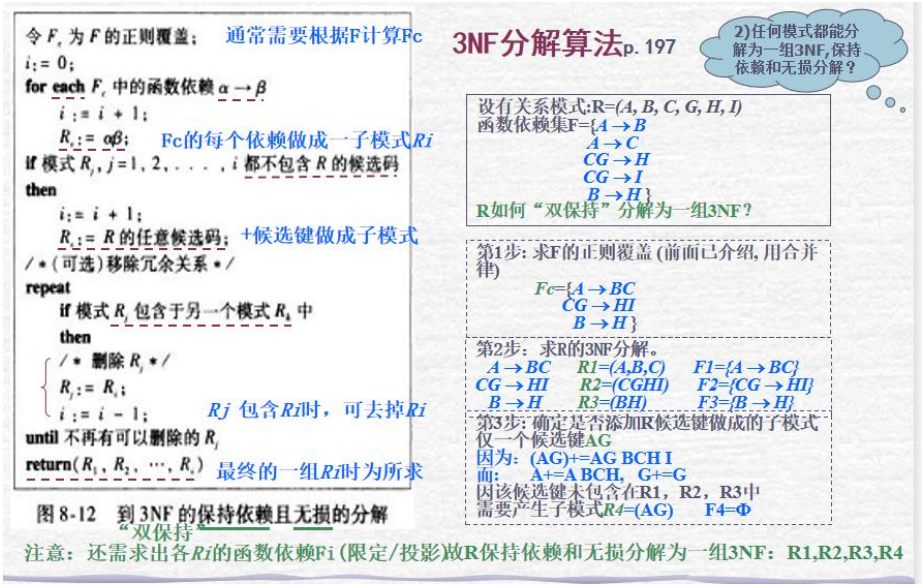
  寻找无关属性。发现{XY→PWZ}中P属性。删除P属性后，F'={Z→X，X→P，XY→WZ}，计算XY+ = XYWZP，包含P属性，故P属性无关，可以删除

  Fc={Z→X，X→P，XY→WZ}中，每个元素都不是无关元素，无法再删，故解得正则覆盖

- 分解算法
  - 1. 首先求 F 的**正则覆盖**
  - 2. **为每个函数依赖 X->A 建一个关系**，关系模式为 XA

    这样确保了函数依赖
  - 3. **确定候选键是否被添加进来**，没有的话要添加候选键的关系模式

    这样确保了无损分解
  - 4. 去重
  - 举例



- 如何判定候选码
  - 算法

设关系模式R中U=ABC.....等N个属性，U中的属性在FD中有四种范围：

(1)左右出现;
(2)只在左部出现;
(3)只在右部出现;
(4)不在左右出现;

算法：按以下步骤求候选键：
1.只在FD右部出现的属性，不属于候选码;
2.只在FD左部出现的属性，一定存在于某候选码当中;
3.外部属性一定存在于任何候选码当中;
4.其他属性逐个与2,3的属性组合，求属性闭包，直至X的闭包等于U,若等于U,则X为候选码。

例1：R<U,F>,U=(A,B,C,D,E,G),F={AB-->C,CD-->E,E-->A.A-->G},求候选码。

因G只在右边出现,所以G一定不属于候选码;而B,D只在左边出现,所以B,D一定属于候选码;BD的闭包还是BD,则对BD进行组合,除了G以外,BD可以跟A,C,E进行组合

先看ABD

ABD本身自包ABD,而AB-->C,CD-->E,A-->G,所以ABD的闭包为ABDCEG=U

再看BDC

CD-->E,E-->A,A-->G,BDC本身自包,所以BDC的闭包为BDCEAG=U

最后看BDE

E-->A,A-->G,AB-->C,BDE本身自包,所以BDE的闭包为BDEAGC=U

因为(ABD)、(BCD)、(BDE)的闭包都是ABCDEG所以本问题的候选码有3个分别是ABC、BCD和BDE