关系数据库理论——数据依赖

数据依赖是一个关系内部属性与属性之间的一种约束关系，通过属性间值的相等与否体现出来的数据间相互联系。

- 主要类型为函数依赖（FD）和多值依赖（MVD）。
    - 函数依赖又可以分为平凡函数依赖与非平凡函数依赖、完全函数依赖与部分函数依赖、传递函数依赖

函数依赖

- 定义：设$R(U)$是一个属性集*U*上的关系模式，*X*和*Y*是*U*的子集。若对于*R(U)*的任意一个可能的关系*r*，*r* 中不可能存在两个元组在*X*上的属性值相等， 而在*Y*上的属性值不等， 则称“*X*函数确定*Y* ”或“*Y*函数依赖于*X* ”，记作$X \to Y$。
- 若$X \to Y$，X**是决定因素**.

  若$X \to Y$，并且$Y \to X$, 则记为$X \longleftrightarrow Y$。

  若Y不函数依赖于X, 则记为$X \nrightarrow Y$。

（非）平凡函数依赖

对于任一关系模式，平凡函数依赖都是必然成立的，它不反映新的语义。若不特别声明， 我们总是讨论非平凡函数依赖。

- $X \to Y$，但$Y \nsubseteq X$则称$X \to Y$是非平凡的函数依赖（ NON-Trivial FD） 。
- $X \to Y$，但$Y \subseteq X$ 则称$X \to Y$是平凡的函数依赖（ Trivial FD） 。

完全（部分）函数依赖

- 定义：在R(U)中，如果$X \to Y$，并且对于X的任何一个真子集X', 都有$X' \nrightarrow Y$, 则称Y对X完全函数依赖，记作$X \to Y$。
- 定义：若$X \to Y$，但Y不完全函数依赖于X，则称Y对X部分函数依赖，记作$X \to Y$。

传递函数依赖

- 定义：在$R(U)$中，如果$X \to Y(Y \nsubseteq X)$，$Y \nrightarrow X$，$Y \to Z$，$Z \nsubseteq Y$, 则称Z对X传递函数依赖(transitive functional dependency)。记为：$X \xrightarrow{传递} Z$。

确定函数依赖

如果X是关系R的超码，则关系R任意子集都存在$X \to Y$。

- 确定函数依赖的方法

- 分析属性间的语义关系

  Addresses(City, Street, Zipcode)

  Zipcode ⟶ City

- 从已知的数据依赖推导出新的依赖

Let R(A, B, C), F = {A⟶B, B⟶C}.

A ⟶C can be derived from F.

A→C 为*F*所蕴涵

基于F所蕴含的关系，可以确定所有的函数依赖，可以确定码。

- 函数依赖闭包

- 在关系模式R<U,F>中为F所逻辑蕴涵的函数依赖的全体叫作F的闭包（closure），记为F^+。

- A BIG F^+ may be derived from a small F.

Example: For R(A, B, C) and

F = {A⟶B, B⟶C}

F^+ = { A⟶B, B⟶C, A⟶C,

A⟶A, B⟶B, C⟶C,

AB⟶AB, AB⟶A, AB⟶B, ... }

F^+是个很大的集合，如何推理？Armstrong公理

- 计算F^+的理论

Armstrong公理系统(1974):

- A1 自反律（reflexivity rule）：若$Y \subseteq X \subseteq U$，则$X \rightarrow Y$为$F$所蕴涵。
- A2 增广律（augmentation rule）：若$X \rightarrow Y$为F所蕴涵，且$Z \subseteq U$，则$XZ \rightarrow YZ$为F所蕴涵。
- A3 传递律（transitivity rule）：若$X \rightarrow Y$及$Y \rightarrow Z$为F所蕴涵，则$X \rightarrow Z$为F所蕴涵。

❖ 根据Armstrong公理系统三条推理规则可以得到下面三条推理规则：

- 合并规则（union rule）：

  由$X \rightarrow Y$，$X \rightarrow Z$，有$X \rightarrow YZ$。
- 伪传递规则（pseudo transitivity rule）：

  由$X \rightarrow Y$，$WY \rightarrow Z$，有$XW \rightarrow Z$。
- 分解规则（decomposition rule）：

  由$X \rightarrow Y$及$Z \subseteq Y$，有$X \rightarrow Z$。

Armstrong公理系统是有效的、完备的

- 属性的闭包

- 在关系模式$R<U,F>$中为F所逻辑蕴涵的函数依赖的全体叫作F的闭包，记为F^+。

- 设F为属性集U上的一组函数依赖，X、$Y \subseteq U$，$X_F^+=\{A \mid X \to A$能由F根据Armstrong公理导出$\}$，X_F^+称为**属性集X关于函数依赖集F的闭包。**

$$X^+ = \{ A \mid X \longrightarrow A \in F^+ \}$$

Theorem: $X \longrightarrow Y \in F^+$ if and only if $Y \subseteq X^+$.

- 求闭包的算法（算法6.1）

- 求属性集X（$X \subseteq U$）关于U上的函数依赖集F的闭包X_F^+

① 初始化：令$X^{(0)}=X$，$i=0$

② 求B：对$X^{(i)}$中的每个元素，依次检查相应的函数依赖,将依赖它的属性加入B。

③ 并：$X^{(i+1)}=B \cup X^{(i)}$。

④ 判断：$X^{(i+1)}= X^{(i)}$。

⑤ 若$X^{(i+1)}$与$X^{(i)}$相等或$X^{(i)}=U$，则$X^{(i)}$就是X_F^+，算法终止。

⑥ 若否，则$i=i+1$，返回第②步。

[Example] Relation : $R<U, F>$

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

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

Question: $(AB)_F^+$?

Solution: Let $X^{(0)}=AB$;

(1) $X^{(1)}=AB\cup CD=ABCD$。

(2) $X^{(0)}\neq X^{(1)}$

$X^{(2)}=X^{(1)}\cup E=ABCDE$。

(3) $X^{(2)}=U$, End

→ $(AB)_F^+=ABCDE$。

求 $(X)_F^+$ 的作用？ 确定码

定理: 如果有$R(A_1, ..., A_n)$ 和函数依赖F in R, $K \subseteq R$ 是

- 超码（ superkey ）如果满足$K^+ = \{A_1, ..., A_n\}$;
- 候选码 （candidatekey）如果 K是超码，并且对于K的任意子集 X, $X^+ \neq \{A_1, ..., A_n\}$.

模式分解

模式分解分为无损链接分解和保持函数依赖分解。

关于模式分解的几个重要事实是：

（1） 若要求分解保持函数依赖，那么模式分离总可以达到 3NF，但不一定能达到BCNF.

（2） 若要求分解既保持函数依赖，又具有无损连接性，可以达到 3NF，但不一定能达到 BCNF。

（3） 若要求分解具有无损连接性，那一定可达到 4NF。

无损链接分解

判定方法1

判定一个分解是否无损连接性

适用于分解为两个关系模式

定理（同定理6.5）: Let R be a relation schema and F be a set of FDs in R. Then a decomposition of R, {R1, R2}, is a lossless-join decomposition if and only if

- $R1 \cap R2 \longrightarrow R1 - R2$; or
- $R1 \cap R2 \longrightarrow R2 - R1$.

1. 计算 R1 ∩ R2 指属性的交集
2. 计算 R1 - R2 属性的差
3. 判定函数依赖关系是否成立

例子

Example

- F={动物名称→动物属性,动物属性→动物居住地,动物名称→动物居住地}
  - **T1 (动物名称，动物居住地)**
  - **T2 (动物属性，动物居住地)**
  - T1 ∩T2=动物居住地
  - T1 -T2=动物名称
  - T2 –T1=动物属性
    - 动物居住地 $\not\longrightarrow$ 动物名称
    - 动物居住地 $\not\longrightarrow$ 动物属性

有损连接分解！

☐Example

☐F={动物名称→动物属性,动物属性→动物居住地,动物名称→动物居住地}

- ■**T1 (动物名称，动物属性)**
- ■**T2 (动物名称，动物居住地)**
- ■ T1 ∩T2=动物名称
- ■T1 -T2=动物属性
- ■T2 –T1=动物居住地
  - 动物名称——→动物属性
  - 动物名称——→动物居住地

无损连接分解！

☐Example

☐F={动物名称→动物属性,动物属性→动物居住地,动物名称→动物居住地}

- ■**T1 (动物名称，动物属性)**
- ■**T2 (动物属性，动物居住地)**
- ■ T1 ∩T2=动物属性
- ■T1 -T2=动物名称
- ■T2 –T1=动物居住地
  - 动物属性 —×→ 动物名称
  - 动物属性——→动物居住地

无损连接分解！

判定方法2

判定一个分解是否无损连接性

适用于分解为多个关系模式

6.2 算法　判定无损连接性

输入：关系模式$R(A_1, A_2, \ldots, A_n)$,它的函数依赖集F以及分解$\rho=\{R_1, R_2, \ldots, R_k\}$。

方法：

（1）构造表：构造一个k行n列的表，第i行对应于关系模式R_i，第j列对应于属性A_j。

（2）填表（根据属性的分配）：如果$A_j \in R_i$，则在第i行第j列上放符号a_j,否则放符号b_{ij}。

（3）更新表（根据F更新）：逐一检查F中的每一个函数依赖，并修改表中的元素。方法：取F中一个函数依赖$X \rightarrow Y$，在X的列中寻找相同的行，然后将这些行中Y的分量改为相同的符号，如果其中有a_j,则将b_{ij}改为a_j;若其中无a_j，则改为某一个b_{ij}。

（4）循环更新：反复检查第（2）步，至无改变为止.

（5）判断：若存在某一行为$a_1, a_2, \ldots, a_k$, 则分解ρ具有无损连接性；如果F中所有函数依赖都不能再修改表中的内容，且没有发现这样的行，则分解ρ不具有无损连接性。

例子

- 举例：已知R<U,F>，U={A,B,C,D,E}，F={A→C,B→C,C→D,DE→C,CE→A}，R的一个分解为R1(AD)，R2(AB)，R3(BE)，R4(CDE)，R5(AE)，判断这个分解是否具有无损连接性。

- ① 构造一个初始的二维表，若“属性”属于“模式”中的属性，则填aj，否则填bij。
- 分解为R1(AD)，R2(AB)，R3(BE)，R4(CDE)，R5(AE)

模式 \ 属性	A	B	C	D	E
R_1(AD)	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2(AB)	a_1	a_2	b_{23}	b_{24}	b_{25}
R_3(BE)	b_{31}	a_2	b_{33}	b_{34}	a_5
R_4(CDE)	b_{41}	b_{42}	a_3	a_4	a_5
R_5(AE)	a_1	b_{52}	b_{53}	b_{54}	a_5

- ② 根据A→C，对上表进行处理，由于属性列A上第1、2、5行相同均为a1，所以将属性列C上的b13、b23、b53改为同一个符号b13（取行号最小值）。

模式 \ 属性	A	B	C	D	E
R_1(AD)	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2(AB)	a_1	a_2	b_{13}	b_{24}	b_{25}
R_3(BE)	b_{31}	a_2	b_{33}	b_{34}	a_5
R_4(CDE)	b_{41}	b_{42}	a_3	a_4	a_5
R_5(AE)	a_1	b_{52}	b_{13}	b_{54}	a_5

- ③ 根据B→C，对上表进行处理，由于属性列B上第2、3行相同均为a2，所以将属性列C上的b13、b33改为同一个符号b13（取行号最小值）。

模式 \ 属性	A	B	C	D	E
R_1(AD)	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2(AB)	a_1	a_2	b_{13}	b_{24}	b_{25}
R_3(BE)	b_{31}	a_2	b_{13}	b_{34}	a_5
R_4(CDE)	b_{41}	b_{42}	a_3	a_4	a_5
R_5(AE)	a_1	b_{52}	b_{13}	b_{54}	a_5

□④ 根据C→D，对上表进行处理，由于属性列C上第1、2、3、5行相同均为b13，所以将属性列D上的值均改为同一个符号a4。

模式 \ 属性	A	B	C	D	E
R_1(AD)	a_1	b_{12}	b_{13}	a_4	b_{15}
R_2(AB)	a_1	a_2	b_{13}	a_4	b_{25}
R_3(BE)	b_{31}	a_2	b_{13}	a_4	a_5
R_4(CDE)	b_{41}	b_{42}	a_3	a_4	a_5
R_5(AE)	a_1	b_{52}	b_{13}	a_4	a_5

□⑤ 根据DE→C，对上表进行处理，由于属性列DE上第3、4、5行相同均为a4a5，所以将属性列C上的值均改为同一个符号a3。

模式 \ 属性	A	B	C	D	E
$R_1(AD)$	a_1	b_{12}	b_{13}	a_4	b_{15}
$R_2(AB)$	a_1	a_2	b_{13}	a_4	b_{25}
$R_3(BE)$	b_{31}	a_2	a_3	a_4	a_5
$R_4(CDE)$	b_{41}	b_{42}	a_3	a_4	a_5
$R_5(AE)$	a_1	b_{52}	a_3	a_4	a_5

□⑥ 根据CE→A，对上表进行处理，由于属性列CE上第3、4、5行相同均为a3a5，所以将属性列A上的值均改为同一个符号a1。

模式 \ 属性	A	B	C	D	E
$R_1(AD)$	a_1	b_{12}	b_{13}	a_4	b_{15}
$R_2(AB)$	a_1	a_2	b_{13}	a_4	b_{25}
$R_3(BE)$	a_1	a_2	a_3	a_4	a_5
$R_4(CDE)$	a_1	b_{42}	a_3	a_4	a_5
$R_5(AE)$	a_1	b_{52}	a_3	a_4	a_5

□存在某一行为$a_1,a_2,\ldots,a_k$, 则分解具有无损连接性

保持函数依赖分解

- 定义：对关系 R 和函数依赖F, 分解 {R1, R2, ..., Rn} 保持函数依赖,如果满足 $F+ = (F1 \cup F2 \cup \ldots \cup Fn) + where Fi = \pi_{Ri}(F), i = 1, \ldots, n.$

判断方法

判定是否保持函数依赖分解

Algorithm DP

Input: A relation schema R, A set of FDs F in R, a decomposition $\{R_1, R_2, ..., R_n\}$ of R.

for every $X \longrightarrow Y \in F$

① if $\exists R_i$ such that $XY \subseteq R_i$

then $X \longrightarrow Y$ is preserved;

② else use Algorithm XYGP to find W;

if $Y \subseteq W$ then $X \longrightarrow Y$ is preserved;

if every $X \longrightarrow Y$ is preserved

then $\{R_1, ..., R_n\}$ is dependency-preserving;

else $\{R_1, ..., R_n\}$ is not dependency-preserving;

Algorithm XYGP

W := X;

repeat for i from 1 to n do

$W := W \cup ((W \cap R_i)^+ \cap R_i)$;

在每个分解后的关系R_i中寻找X可以确定的属性集

until there is no change to W;

例子

示例: Suppose R(A, B, C, D),

$F = \{A \longrightarrow B, B \longrightarrow C, C \longrightarrow D, D \longrightarrow A\}$,

R1(A,B), R2(B,C), R3(C,D).

Is {R1, R2, R3} dependency-preserving?

Since $AB \subseteq R1$, $A \longrightarrow B$ is preserved.

Since $BC \subseteq R2$, $B \longrightarrow C$ is preserved.

Since $CD \subseteq R3$, $C \longrightarrow D$ is preserved.

For $D \longrightarrow A$, use Algorithm XYGP to compute W.

Initialization: $W = D$;

first iteration:

$W = D \cup ((D \cap AB)^+ \cap AB) = D$;

$W = D \cup ((D \cap BC)^+ \cap BC) = D$;

$W = D \cup ((D \cap CD)^+ \cap CD)$

$= D \cup (D^+ \cap CD)$

$= D \cup (ABCD \cap CD) = CD$;

second iteration:

$W = CD \cup ((CD \cap AB)^+ \cap AB) = CD;$

$W = CD \cup ((CD \cap BC)^+ \cap BC)$

$= CD \cup (C^+ \cap BC) = BCD;$

$W = BCD \cup ((BCD \cap CD)^+ \cap CD)$

$= BCD;$

third iteration:

$W = BCD \cup ((BCD \cap AB)^+ \cap AB)$

$= ABCD;$

Since $A \subseteq W$, $D \longrightarrow A$ is also preserved.

Hence, {R1, R2, R3} is a dependency-preserving decomposition.

范式

- 范式是符合某一种级别的关系模式的集合。
- 范式分为如下几种：1NF、2NF、3NF和BCNF。它们之间的关系为
  $1NF \supset 2NF \supset 3NF \supset BCNF \supset 4NF \subset 5NF$
  - 1NF：数据库表的每一列都是不可分割的基本数据项，即实体中的某个属性不能有多个值或者不能有重复的属性。第一范式（1NF）是对关系模式的基本要求，不满足第一范式（1NF）的数据库就不是关系数据库。
  - 2NF：若关系模式$R \in 1NF$，并且每一个**非主属性**都**完全函数依赖**于**任何**一个**候选码**，则$R \in 2NF$。

Sid	Sname	Phone	Course-id	Course-description	Credit-hours	Grade
100	John	487 2454	IS380	Database Concepts	3	A
100	John	487 2454	IS416	Unix Operating System	3	B
200	Smith	671 8120	IS380	Database Concepts	3	B
200	Smith	671 8120	IS416	Unix Operating System	3	B
200	Smith	671 8120	IS420	Data Net Work	3	C
300	Russell	871 2356	IS417	System Analysis	3	A

(Sid, Course-id) 是候选码

*

❖ 一个关系模式不属于2NF，会产生以下问题：

- 插入异常
- 删除异常
- 修改复杂

❖ 解决方法：按照非主属性对主属性的依赖关系分解（如上页图）

- 用投影分解把关系模式分解成三个关系模式
- 根据属性函数依赖关系进行分解
  - Student (Sid:pk, Sname, Phone)
  - Courses (Course-id:pk, Course-Description, Credit-hours)
  - **Student-grade (Sid:pk1:fk:Student, Course-id::pk2:fk:Courses,** Grade)

- 3NF：设关系模式R<U,F>∈1NF,若R中不存在这样的码X、属性组Y及非主属性Z（$Z \notin Y$），使得X→Y，Y→Z成立，Y ↛ X，则称R<U,F> ∈ 3NF。上述定义分为两种情况：$X \notin Y$，（如 X→Y，Y→Z成立，部分函数依赖）。$X \in Y$，（如 X→Y，Y→Z成立，传递函数依赖）。若R<U,F> ∈ 3NF，则每一个非主属性即不传递依赖于码，又不部分依赖于码。 2NF + 非主属性对码无传递函数依赖。

Sid	Activity	Fee
100	Swimming	100
200	Tennis	100
300	Golf	300
400	Swimming	100

Student ID (SID)---------->Activity I,e SID determine Activity
Further Activity ----------> fee that is the Activity determine the fee

❖ 上述关系属于2NF

❖ 但是有传递函数依赖，不属于3NF

分解：从有传递的属性处断开成2个关系

- BCNF：BCNF（Boyce Codd Normal Form）是由Boyce 与 Codd 提出的，比上述的3NF 又造一步，通常认为 BCNF 是修正的第三范式，有时也称为扩充的第三范式。关系模式 $R<U,F>\in 2NF$，若 $X \to Y$且$Y \notin X$时X必含有码，则$R<U,F>\in BCNF$也就是说，关系模式 R<U,F>中，若每一个决定因素都包含码，则$R<U,F>\in BCNF$

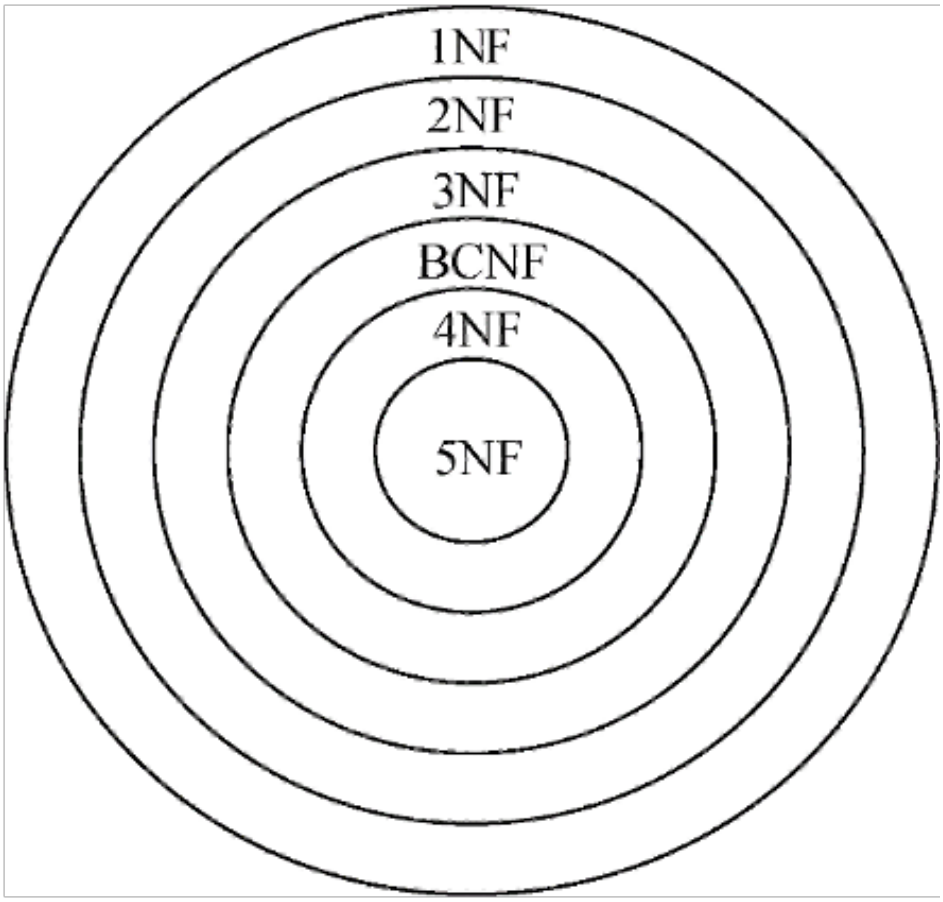
  由BCNF 的定义可以得到结论，一个满足 BCNF 的关系模式有：

  • 所有非主属性对每一个码都是完全函数依赖。

•所有主属性对每一个不包含它的码也是完全函数依赖。

•没有任何属性完全函数依赖于非码的任何一组属性。

如果一个关系数据库中的所有关系模式都属于BCNF，那么在函数依赖范畴内，它已实现了模式的彻底分解，达到了最高的规范化程度，消除了插入异常和删除异常.



- 一个低一级范式的关系模式，通过模式分解可以转化为若干个高一级范式的关系模式的集合，这个过程称为**规范化**。规范化的目的是减少数据冗余和消除异常（更新异常、插入异常、删除异常）。

确定关系的范式

- 首要任务：确定函数依赖和候选码
- 然后根据上述范式定义得到。

小结

- 不能说规范化程度越高的关系模式就越好。
  - 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。

最小函数依赖

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，亦称为最小依赖集或最小覆盖。

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。即F中的函数依赖均不能由F中其他函数依赖导出。

3. F中不存在这样的函数依赖X→A，X有真子集Z使得$F-X\rightarrow A\cup Z\rightarrow A$与F等价。 即F中各函数依赖左部均为最小属性集（不存在冗余属性）。

求解方法

1. 用分解的法则，使F中的任何一个函数依赖的右部仅含有一个属性；
2. 去掉多余的函数依赖：从第一个函数依赖X→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→Y；否则不能去掉，依次做下去。直到找不到冗余的函数依赖；
3. 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→A，若要判Y为多余的，则以X→A代替XY→A是否等价？若$A\in(X)^{+}$，则Y是多余属性，可以去掉。

规范化到BCNF（LLJD-BCNF）

Algorithm LLJD-BCNF

Input: A relation schema R, a set of FDs in R.

Output: A lossless-join decomposition D such that each new schema in D is in BCNF.

```
D := {R};
while ∃ Rᵢ ∈ D that is not in BCNF do {
    find an FD X ⟶ Y such that
      (1) X ⟶ Y makes Rᵢ not in BCNF, and
      (2) XY ⊆ Rᵢ;
    replace Rᵢ by (Rᵢ – Y) and XY in D;
}
```

规范化到3NF（LLJD-DPD-3NF）

Algorithm LLJD-DPD-3NF

Input: A relation schema R, a set of FDs F in R.

Output: A lossless-join and dependency-preserving decomposition D such that each schema in D is in 3NF.

（1）找出所有候选键。

（2）计算最小函数依赖集(F_{min}) 。

（3）令 ($X \to A_i$) （$(i = 1, \cdots, m)$）为 (F_{min}) 中所有左边相同为 (X) 的函数依赖。如果 ($X \cup A_1 \cup \cdots \cup A_m$) 不是现有关系模式的子模式，则在 (D) 中创建一个关系模式 ($X \cup A_1 \cup \cdots \cup A_m$) 。

（4）如果这些关系模式中没有一个包含 (R) 的候选键，则使用一个候选键来形成另一个关系模式。