数据库系统 期末速通教程

5. 关系型数据库理论

5.1 Determinacy

5.1.1 Determinacy

[**Definition 5.1.1.1**] [**Anomalies**]

(1) [**update anomalies**] changing information in one tuple leaves the same information unchanged in another; occurs with redundancy.

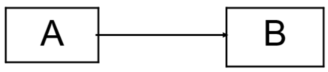
(2) [**deletion anomalies**] if a set of values becomes empty, we may lose other information as a side effect.

(3) [**insertion anomalies**] inability to represent certain information.

(4) solution: remove redundancy by restructuring the DB schema using **normalization**.

[**Definition 5.1.1.2**] [**Determinacy**] Certain attributes can be used to uniquely determine one value of another attribute. A is a **determinant** of B if each value of A has precisely one (possibly null) value of B associated with it. We say, A **determines** B or B is **functionally dependent** on A .

Produce a simple diagram to show A determines B :

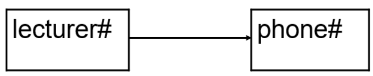


[**Example 5.1.1.1**]

(1) Each lecturer has one phone number. Each phone number may be associated with many lecturers, e.g. shared offices or communal phones.

① lecturer# determines phone#.

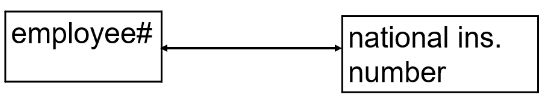
② phone# doesn't determine lecturer#.



(2) Each employee has one national insurance number. Each national insurance number is allocated to exactly one employee.

① employee# determines national insurance number#.

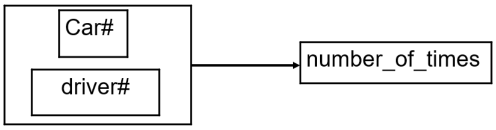
② national insurance number# determines employee#.



[**Definition 5.1.1.3**] [**Composite Determinacy**] Several attribute values taken together can determine the value of some other attribute.

[**Example 5.1.1.2**] Driver# can have several number_of_times#. Car# can have several number_of_times#.

Driver# and Car# has one number_of_times#, gives (Driver#, Car#) determines number_of_times#.



5.1.2 Functional Dependency

[**Definition 5.1.2.1**] [**Functional Dependency**, **FD**] Let $R(A_1, \cdots, A_n)$ be a relation schema. Let X and Y be two subsets of $\{A_1, \cdots, A_n\}$. X is said to **functionally determine** Y or Y is **functionally dependent** on X if for every legal relation instance $r(R)$, for any two tuples t_1 and t_2 in $r(R)$, we have $t_1[x] = t_2[x] \Rightarrow t_1[Y] = t_2[Y]$.

$X \subseteq R$ denotes that X is a subset of the attributes of R , $X \to Y$ denotes that X is functionally determines Y .

$X \to Y$ in R

$\Leftrightarrow$ for $\forall t_1, t_2 \in r(R)$, if t_1 and t_2 have the same X - value, then t_1 and t_2 have the same Y - value.

$\Leftrightarrow$ there exists no $t_1, t_2 \in r(R)$ s.t. t_1 and t_2 have the same X - value but different Y - values.

$\Leftrightarrow$ for each X - value, there corresponds to a unique Y - value.

[**Example 5.1.2.1**] Identify functional dependency.

(1) Created by assertions.

Employees(SSN, Name, Years_of_emp, Salary, Bounus) .

Assertion: Employees hired the same year have the same salary.

Implies that: Years_of_emp $\to$ Salary.

(2) Analyze the semantics of attributes of R .

Addresses(City, Street, Zipcode) .

Implies that: Zipcode $\to$ City.

(3) Derive new FDs from existing FDs.

Let $R(A, B, C)$ and $F = \{A \to B, B \to C\}$, than $A \to C$ can be derived from F .

Denote F **logically implies** $A \to C$ by $F \models (A \to C)$.

[**Theorem 5.1.2.1**]

(1) If X is a superkey of R and Y is any subset of R , then $X \to Y$ in R .

(2) If $Y \subseteq X \subseteq R$, then $X \to Y$. Specially, $A \to A$ for any attribute A .

[**Definition 5.1.2.2**] [**Closure**] Let F be a set of FDs in R . The closure of F is the set of all FDs that are logically implied by F , denoted by F^+ , i.e. , $F^+ = \{X \to Y \mid (F \models X \to Y)\}$.

[**Theorem 5.1.2.2**] [**Armstrong's Axioms**]

(1) [**自反律**, **Reflexivity Rule**] If $X \supseteq Y$, then $X \to Y$.

(2) [**扩充律**, **Augmentation Rule**] $\{X \to Y\} \models (XZ \to YZ)$.

(3) [**传递律**, **Transitivity Rule**] $\{X \to Y, Y \to Z\} \models X \to Z$.

[**Note 1**] Armstrong's Axioms are **sound** and **complete**.

① sound: No incorrect FD can be generated from F using Armstrong's Axioms.

② complete: Given a set of FDs F , all FDs in F^+ can be generated using Armstrong's Axioms.

[**Corollary 1**]

(1) [**分解律**, **Decomposition Rule**] $\{X \to YZ\} \models \{X \to Y, X \to Z\}$.

(2) [**结合律**, **Union Rule**] $\{X \to Y, X \to Z\} \models (X \to YZ)$.

(3) [**伪传递律**, **Pseudo-transitivity Rule**] $\{X \to Y, WY \to Z\} \models (WX \to Z)$.

[**Proof**]

(1) According to Reflexivity Rule , we have $YZ \to Y$, $YZ \to Z$.

$X \to YZ$, $YZ \to Y$, according to Transitivity Rule, we have $X \to Y$.

Similarly, we have $X \to Z$.

(2) $X \to Y$, according to Augmentation Rule, we have $XX \to XY$, i.e. , $X \to XY$.

Similarly, we have $XY \to ZY$. According to Transitivity Rule, we have $X \to YZ$.

(3) $X \to Y$, according to Augmentation Rule, we have $XW \to YW$.

$WY \to Z$, according to Transitivity Rule, we have $WX \to Z$.

[**Corollary 2**] If $X \subseteq R$ and $A_1, \cdots, A_n$ are attributes in R , then ($X \to A_1 \cdots A_n) \equiv \{X \to A_1, \cdots, X \to A_n\}$.

[**Theorem 5.1.2.3**] Determine whether $F \models (X \to Y)$ is true.

(1) $F \models (X \to Y)$ iff $(X \to Y) \in F^+$.

(2) Denotes the **closure** of X under F as X^+ which is the set of attributes that are functionally determined by X under F , i.e. , $X^+ = \{A \mid (X \to A) \in F^+\}$. $(X \to Y) \in F^+$ iff $Y \subseteq X^+$.

[**Proof**] (2) Use Decomposition Rule and Union Rule.

[**Note 1**] Computing F^+ could be very expensive.

[**Node 2**] Algorithm for computing X^+ :

Input: a set of FDs F , a set of attributes X in R .
Output: X^+ .

begin:
$X^+ \leftarrow X$ **;**
repeat for each FD $Y \to Z$ in F , do:
if $Y \subseteq X^+$, then $X^+ \leftarrow X^+ \bigcup Z$;
until nothing change to X^+ ;
end

The performance of this algorithm is sensitive to the order of FDs in F .

[**Theorem 5.1.2.4**] Given $R(A_1, \cdots, A_n)$ and a set of FDs F in R , then $K \subseteq R$ is a:

(1) superkey if $K^+ = \{A_1, \cdots, A_n\}$.

(2) candidate key if K is a superkey and for any proper subset X of K , $X^+ \neq \{A_1, \cdots, A_n\}$.

[**Example 5.1.2.2**] $R(A, B, C, G, H, I) = ABCGHI$, $F = \{A \to B, CG \to HI, B \to H, A \to C\}$.

(1) Let $X = AG$. Compute X^+ .

(2) Determine whether AG is a candidate key.

[**Solution**]

(1) ①Initialization: $X^+ = AG$.

② 1 st iteration:

(i) Consider $A \to B$, since $A \subseteq X^+$, then $X^+ = ABG$.

(ii) Consider $CG \to HI$, since $CG \nsubseteq X^+$, then nothing change to X^+ .

(iii) Consider $B \to H$, since $B \subseteq X^+$, then $X^+ = ABGH$.

(iv) Consider $A \to C$, since $A \subseteq X^+$, then $X^+ = ABCGH$.

Now $X^+ = ABCGH$.

③ 2 nd iteration:

(i) Consider $A \to B$, since $A \subseteq X^+$, then $X^+ = ABCGH$.

(ii) Consider $CG \to HI$, since $CG \subseteq X^+$, then $X^+ = ABCGHI$.

(iii) Consider $B \to H$, since $B \subseteq X^+$, then $X^+ = ABCGHI$.

(iv) Consider $A \to C$, since $A \subseteq X^+$, then $X^+ = ABCGHI$.

Now $X^+ = ABCGHI$.

④ 3 rd iteration: Consider each FD in F again, but nothing change to X^+ , exit.

Result: $X^+ = ABCGHI$.

(2) AG is a superkey since $(AG)^+ = ABCGHI$.

Neither A nor G is a superkey since $A^+ = ABCH$, $G^+ = G$.

Hence, AG is a candidate key and can be used as primary key.

[**Theorem 5.1.2.5**] Finding candidate keys from FDs.

Let F be a set of FDs in relation schema $R(A_1, \cdots, A_n)$.

(1) [**Brute Force**]

① For each $A_i \ \ (i = 1, \cdots, n)$, compute A_i^+ . If $A_i^+ = A_1 \cdots A_n$, then A_i is a candidate key.

② For each pair $A_iA_j \ \ (1 \leq i, j \leq n, i \neq j)$:

(i) If A_i or A_j is a candidate key, then A_iA_j is not a candidate key.

(ii) Otherwise, compute $(A_iA_j)^+$. If $(A_iA_j)^+ = A_1 \cdots A_n$, then A_iA_j is a candidate key.

③ For each triple $A_iA_jA_k \ \ (1 \leq i, j, k \leq n, i \leq j, j \leq k)$: ...

④ ...

(2) [**Graph Approach**]

① Draw the dependency graph of F . Each vertex corresponds to an attribute. Edges are defined as follows:

$A \to B$	$A \to BC$	$AB \to C$
$\begin{tikzcd} A \arrow[r] & B \end{tikzcd}$	$\begin{tikzcd} A \arrow[r] \arrow[dr] & B \\ & C \end{tikzcd}$	$\begin{tikzcd} A \arrow[dr, no head] & & \\ & {} \arrow[r] & C \\ B \arrow[ur, no head] & & \end{tikzcd}$

② Identify the set of vertices V_{ni} that have no incoming edges.

Claim 1: Any candidate key must have all attributes in V_{ni} .

Claim 2: If V_{ni} forms a candidate key, then it's the only candidate key.

③ Identify the set of vertices V_{oi} that have only incoming edges.

Claim 3: No candidate key will contain any attribute in V_{oi} .

④ Use the observation to find other candidate keys if exist.

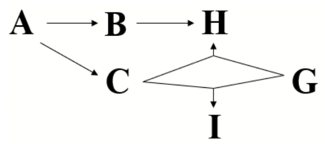
[**Example 5.1.2.3**] Find all candidate keys.

(1) $R(A,B,C,G,H,I)$, $F=\{A\to BC, CG\to HI, B\to H\}$.

(2) $R(A,B,C,D,E,H)$, $F=\{A\to B, AB\to E, BH\to C, C\to D, D\to A\}$.

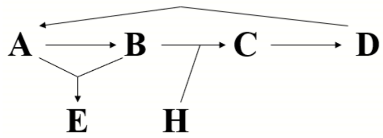
[**Solution**]

(1) $V_{\text{ni}}=\{A,G\}$, $V_{\text{oi}}=\{H,I\}$.



Since $(AG)^+=ABCGHI$, then AG is the only candidate key of R .

(2) $V_{\text{ni}}=\{H\}$, $V_{\text{oi}}=\{E\}$.



Candidate keys: AH 、 BH 、 CH 、 DH .

5.1.3 Relation Decomposition

[**Definition 5.1.3.1**] [**Decomposition**] Let R be a relation schema. A set of relation schemas $\{R_1,\cdots,R_n\}$ is a **decomposition** of R if $R=R_1\bigcup\cdots\bigcup R_n$.

[**Note 1**] If $\{R_1,\cdots,R_n\}$ is a decomposition of R , r is an instance of R , then $r\subseteq\pi_{R_1}(r)\infty\ \cdots\ \infty\pi_{R_n}(r)$.

[**Note 2**] Information may be lost (i.e. wrong tuples may be added) due to a decomposition.

[**Definition 5.1.3.2**] [**Lossless Join Decomposition**, **LLJD**] $\{R_1,\cdots,R_n\}$ is a **lossless** (non-additive) **join decomposition** of R if for every legal instance r of R , $r=\pi_{R_1}(r)\infty\ \cdots\pi_{R_n}(r)$.

[**Theorem 5.1.3.1**] Let R be a relation schema and F be a set of FDs in R , $\{R_1,R_2\}$ is a decomposition of R . $\{R_1,R_2\}$ is a LLJD iff $R_1\bigcap R_2\to R_1\backslash R_2$ or $R_1\bigcap R_2\to R_2\backslash R_1$.

[**Example 5.1.3.1**] $\text{Prod\_Manu}(\text{Prod\_no},\text{Prod\_name},\text{Price},\text{Manu\_id},\text{Manu\_name},\text{Address})$, $F=\{\text{P\#}\to\text{PnPrMid},\text{Mid}\to\text{Mn A}\}$. Determine whether $\{\text{Products},\text{Manufacturers}\}$ is a LLJD, where $\text{Products}=\{\text{P\#},\text{Pn},\text{Pr},\text{Mid}\}$, $\text{Manufacturers}=\{\text{Mid},\text{Mn},\text{A}\}$.

[**Solution**] Since $\text{Products}\bigcap\text{Manufactures}=\text{Mid}\subseteq\{\text{Mid},\text{A}\}=\text{Manufactures}\backslash\text{Products}$,

hence $\{\text{Products},\text{Manufacturers}\}$ is a LLJD.

[**Definition 5.1.3.3**] Let R be a relation schema and F be a set of FDs in R. For any $R' \subseteq R$, the **restriction** of F to R' is a set of all FDs F' in F^+ such that each FD in F' contains only attributes in R', i.e., $F' = \pi_{R'}(F) = \{(X \to Y) \mid XY \subseteq R' \wedge F \models (X \to Y)\}$.

[**Note**] $\pi_{R'}(F) = \pi_{R'}(F^+)$.

[**Definition 5.1.3.4**] [**Dependency Preserving Decomposition**, **DPD**] Given a relation schema R and a set of FDs F in R. $\{R_1, \cdots, R_n\}$ is a decomposition of R. $\{R_1, \cdots, R_n\}$ is **dependency preserving** if $F^+ = (F_1 \bigcup \cdots F_n)^+$ where $F_i = \pi_{R_i}(F) \ \ (i = 1, \cdots, n)$.

[**Note**] Algorithm for determining DP.

Input: a relation schema R, a set of FDs F in R, a decomposition $\{R_1, \cdots, R_n\}$ of R.
Output: Determine whether $\{R_1, \cdots, R_n\}$ is DPD.

begin:
for every $(X \to Y) \in F$:
if $\exists\ R_i$ s.t. $XY \subseteq R_i$, then $(X \to Y)$ is preserved;
else:
use Algorithm XYGP to find W;
if $Y \subseteq W$, then $(X \to Y)$ is preserved.

If every $(X \to Y)$ is preserved, then $\{R_1, \cdots, R_n\}$ is DPD;
else, $\{R_1, \cdots, R_n\}$ is not DPD.
end

Algorithm XYGP:

begin:
$W \leftarrow X$;
repeat:
for i from 1 to n, $W \leftarrow W \bigcup \left((W \bigcap R_i)^+ \bigcap R_i\right)$;
until nothing change to W;
end

[**Example 5.1.3.2**] $R(\text{City}, \text{Street}, \text{Zipcode})$, $F = \{\text{CS} \to \text{Z}, \text{Z} \to \text{C}\}$, $R_1(\text{S}, \text{Z})$, $R_2(\text{C}, \text{Z})$.

$\pi_{R_1}(F) = \{\text{S} \to \text{S}, \text{Z} \to \text{Z}, \text{SZ} \to \text{S}, \text{SZ} \to \text{Z}, \text{SZ} \to \text{SZ}\}$.

$\pi_{R_2}(F) = \{\text{Z} \to \text{C}, \text{C} \to \text{C}, \text{Z} \to \text{Z}, \text{CZ} \to \text{C}, \text{CZ} \to \text{Z}, \text{CZ} \to \text{CZ}\}$.

Since $(\text{CS} \to \text{Z}) \in F^+$ but $(\text{CS} \to \text{Z}) \notin (\pi_{R_1}(F) \bigcup \pi_{R_2}(F))^+$, hence $\{R_1, R_2\}$ is not a DPD.

[**Example 5.1.3.3**] $R(A,B,C,D)$, $F=\{A\to B,B\to C,C\to D,D\to A\}$. Determine whether $\{R_1,R_2,R_3\}$ is a DPD, where $R_1(A,B)$, $R_2(B,C)$, $R_3(C,D)$.

[**Solution**]

(1) $(A\to B)$ is preserved since $AB\subset R_1$. Similarly, $(B\to C)$ and $(C\to D)$ are preserved.

(2) For $D\to A$, use algorithm XYGP to compute W .

① Initialization: $W=D$.

② 1 st iteration:

(i) $W=D\bigcup\left((D\bigcap AB)^+\bigcap AB\right)=D$.

(ii) $W=D\bigcup\left((D\bigcap BC)^+\bigcap BC\right)=D$.

(iii) $W=D\bigcup\left((D\bigcap CD)^+\bigcap CD\right)=D\bigcup(D^+\bigcap CD)$

$=D\bigcup(ABCD\bigcap CD)=CD$.

② 2 nd iteration:

(i) $W=CD\bigcup\left((CD\bigcap AB)^+\bigcap AB\right)=CD$.

(ii) $W=CD\bigcup\left((CD\bigcap BC)^+\bigcap BC\right)=CD\bigcup(C^+\bigcap BC)=BCD$.

(iii) $W=BCD\bigcup\left((BCD\bigcap CD)^+\bigcap CD\right)=BCD$.

③ 3 rd iteration: $W=BCD\bigcup\left((BCD\bigcap AB)^+\bigcap AB\right)=ABCD$.

$(D\to A)$ is preserved since $A\subseteq W$.

Hence, $\{R_1,R_2,R_3\}$ is a DPD.

5.2 Normalization (1)

5.2.1 Normalization

[**Definition 5.2.1.1**] [**Normalization**] **Normalization** is a technique for producing a set of relations with desirable properties, given the data requirements of an enterprise. Normalization level: 1 st, 2 nd, 3 rd, Boyce-Codd, 4 th and 5 th **Normal Forms** (**NF**), where strictness trend: 1 NF $\to$ 5 NF.

[**Note 1**]

(1) Data normalization is primarily a tool to validate and improve a logical design so that it satisfies certain constraints that avoid unnecessary duplication of data.

(2) Normalization removes redundancy by table decomposition.

(3) The process of decomposing relations with anomalies to produce smaller and **well-structured relations**.

[**Well-structured Relations**] A relation that contains minimal data redundancy and allows users to insert, delete and update rows without causing data inconsistencies is called **well-structured relations**.

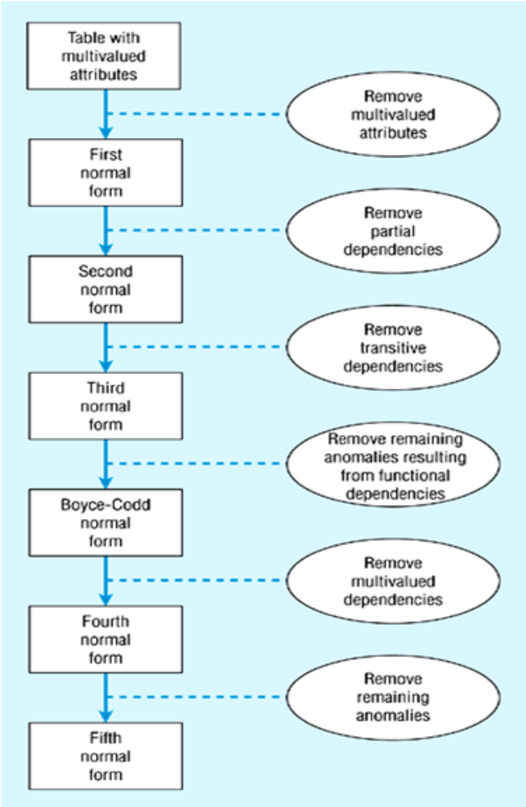
(4) Goal is to avoid anomalies such as:

① [**Insertion Anomaly**] Adding new rows forces user to create duplicate data.

② [**Deletion Anomaly**] Deleting rows may cause a loss of data that would be needed for other future rows.

③ [**Modification Anomaly**] Changing data in a row forces changes to other rows because of duplication.

[**Note 2**] The process of normalization:



5.2.2 1 NF 、 2 NF 、 3 NF 、 BCNF

[**Definition 5.2.2.1**] [**First Normal Form**, **1 NF**]

(1) No multivalued attributes.

(2) Every attribute value is atomic.

(3) Every non-primary-key attribute is functionally dependent on the primary key.

[**Note**] "functionally dependent on the primary key" means:

(1) No multi-value cells.

(2) No nesting relation.

[**Solution**]

(1) Form new relations for each nonatomic atrribute or nested relation along with the PK of the original table.

(2) Choose PK for the newly generated relation.

[**Example 5.2.2.1**]

<u>supplierNo</u>	supplierName	partNo
S5	Wells	P1
S2	Heath	P1
S2	Heath	P4
S7	Barron	P6
S9	Edwards	P8,
S9	Edwards	P2
S9	Edwards	P6

(1) Not in 1 NF since it embeds other relations.

(2) Break $\mathrm{SupplierPart}(\mathrm{supplierNo}, \mathrm{supplierName}, \mathrm{Part}(\mathrm{partNo}))$ down as:

① $\mathrm{Supplier}(\underline{\mathrm{supplierNo}}, \mathrm{supplierName})$.

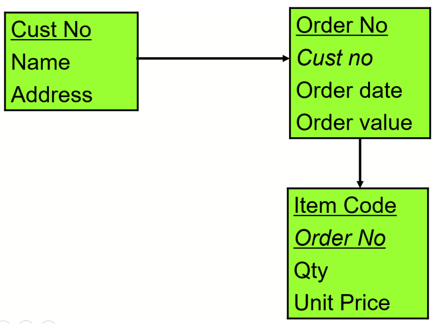
② $\mathrm{SupplierPart}(\underline{\mathrm{supplierNo}}, \mathrm{partNo})$.

[**Example 5.2.2.2**]

```
Customer number
Customer name
Customer address
Order (
  Order no
  Order date
  Order value
  Order item (
    Item code
    Item quantity
    Item unit price
  )
)
```

(1) Not in 1 NF since it embeds other relations.

(2) Break it down as:



[**Definition 5.2.2.2**] [**Second Normal Form**, **2 NF**] A relation is in **Second Normal Form** if :

(1) It's in 1 NF .

(2) Every non-key attribute is **fully functionally dependent** on any key.

[**Fully Functionally Dependent**] B is fully **functionally dependent** on A if B is functionally depend on A , but not on any proper subset of A .

[**Note**]

(1) 2NF = 1 NF + every non-key attribute is fully functionally dependent on the entire PK, that is, Every non-key attribute must be defined by the entire key, not by only part of the key.

(2) No partial functionally dependencies.

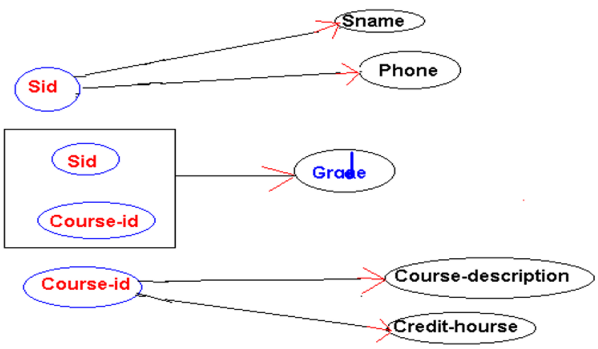
(3) A 1 NF relation with non-composite primary key is in 2 NF.

[**Solution**] Create a set of new relations:

(1) One relation for the attributes that are fully dependent upon the key.

(2) One relation for each part of the key that has partially dependent attributes.

[**Example 5.2.2.3**]



(1) 上图中的学生课程表 "students-courses" 不是 2 NF (上图是一个表).

(2) 将上图拆分为:

① $\text{students}(\text{sid} : \text{PK}, \text{sname}, \text{phone})$.

② $\text{courses}(\text{cid} : \text{PK}, \text{cdescription})$.

③ $\text{students-grade}(\text{sid} : \text{PK} : \text{FK}(\text{students}), \text{cid} : \text{PK} : \text{FK}(\text{courses}), \text{grade})$.

[**Example 5.2.2.4**]

(1) 表 $\text{Result}(\text{sid}, \text{cid}, \text{ctitle}, \text{mark})$ 不是 2 NF .

(2) 将其拆分为:

① $\text{Result}(\underline{\text{sid}}, \underline{\text{cid}}, \text{mask})$.

② $\text{Courses}(\underline{\text{cid}}, \text{ctitle})$.

[**Definition 5.2.2.3**] [**Third Normal Form**, **3 NF**] A relation is in **Third Normal Form** if :

(1) It's in 2 NF.

(2) No non-key attribute is **transitively dependent** on any key.

[**transitively dependent**] If $A \to B$ and $B \to C$, then C is **transitively dependent** on A , which provided that A is not functionally dependent on B or C .

[**Note**]

(1) 3 NF = 2 NF + no transitive dependences.

(2) A 2 NF relation with only one non-key attribute is in 3 NF.

(3) 3 NF relations are sufficient for most practical database design problems but doesn't guarantee that all anomalies have been removed.

(4) 3 NF relations with more than one CK may result in anomalies.

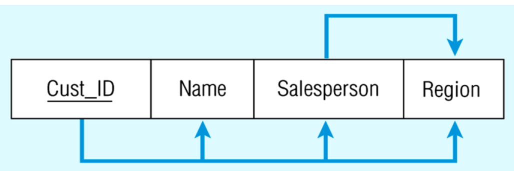
(5) 3 NF does not deal with overlapping candidate keys, i.e. , composite candidate keys with at least one attribute in common.

[**Solution**] Remove the attributes involved in transitive dependencies and put them into a new relation.

[**Example 5.2.2.5**]

SALES

Cust_ID	Name	Salesperson	Region
8023	Anderson	Smith	South
9167	Bancroft	Hicks	West
7924	Hobbs	Smith	South
6837	Tucker	Hernandez	East
8596	Eckersley	Hicks	West
7018	Arnold	Faulb	North



(1) In 2 NF , since

$\text{Cust\_ID} \to \text{Name}$, $\text{Cust\_ID} \to \text{Salesperson}$, $\text{Cust\_ID} \to \text{Region}$, no partially dependencies.

(2) Not in 3 NF , since $\text{Cust\_ID} \to \text{Salesperson} \to \text{Region}$.

(3) Remove a transitive dependency.

SALES1

Cust_ID	Name	Salesperson
8023	Anderson	Smith
9167	Bancroft	Hicks
7924	Hobbs	Smith
6837	Tucker	Hernandez
8596	Eckersley	Hicks
7018	Arnold	Faulb

SPERSON

Salesperson	Region
Smith	South
Hicks	West
Hernandez	East
Faulb	North

[**Example 5.2.2.6**] $\text{Customers}(\underline{\text{cid}}, \text{name}, \text{address}, \text{creditCode}, \text{creditLimit})$.

Assume that one credit code can apply to several customers. But one customer can have only one credit code.

(1) Not in 3 NF , since $\text{cid} \to \text{creditCode} \to \text{creditLimit}$.

(2) Set up a relation that includes non-key attribute(s) that functionally determine(s) other non-key attribute(s).

$\text{Customers}(\text{cid}, \text{name}, \text{address}, \text{creditCode})$.

$\text{Credit}(\text{creditCode}, \text{creditLimit})$.

[**Definition 5.2.2.4**] [**Boyce-Codd Normal Form**, **BCNF**] A relation is in **Boyce-Codd Normal Form** if :

(1) It's in 3 NF .

(2) Every **determinant** is a candidate key.

[**determinant**] A **determinant** is any attribute (simple or composite) on which some other attributes is fully functionally dependent.

[**Note**]

(1) BCNF applies to a relation with more than one CK where CKs are composite and overlapping.

(2) A relation schema is **in good quality** if it's in BCNF .

(3) Relation schema with exactly two attributes is in BCNF .

[**Example 5.2.2.7**] $\text{SSL}(\underline{\text{student}}, \underline{\text{subject}}, \text{lecturer})$.

Assume that each teacher teach only one subject, a subject has several teachers to teach.

(1) Not in BCNF , since $\{\text{student}, \text{subject}\} \to \text{lecturer}$, $\text{lecturer} \to \text{subject}$.

(2) Break down into two tables:

$\text{LS}(\underline{\text{lecture}}, \text{subjecet})$.

$\text{SL}(\underline{\text{student}}, \underline{\text{lecturer}})$.

[**Example 5.2.2.8**] $R(\underline{a}, \underline{b}, c, d)$ with $\{a, c\} \to \{b, d\}$, $\{a, d\} \to b$.

(1) $\{a, c\} \to \{b, d\}$ suggests that RK can be changed from $\{a, b\}$ to $\{a, c\}$.

If done, all of the non-key attributes present in R could still be determinded, therefore the change is legal.

(2) $\{a, d\} \to b$ indicates that a, d determine b , but $\{a, d\}$ is not a key since $\{a, d\}$ does not determine all of the non-key attributes like c , therefore can't be CK .

(3) In 3 NF but not in BCNF .

(4) Break down into two tables:

$R_1(\underline{a}, \underline{c}, d)$.

$R_2(\underline{a}, b, \underline{d})$.

[**Example 5.2.2.9**] $\text{Suppliers}(\underline{\text{sid}}, \underline{\text{pid}}, \text{sname}, \text{qty})$ with $\text{sid} \leftrightarrow \text{sname}$.

(1) Not in BCNF , since $\text{sid} \to \text{sname}$ but sid is not a CK .

(2) Break down into two tables:

① $\text{Suppliers}(\underline{\text{sid}}, \text{sname})$, $\text{Products}(\underline{\text{sid}}, \underline{\text{pid}}, \text{qty})$.

Or

② $\text{Suppliers}(\underline{\text{sname}}, \text{sid})$, $\text{Products}(\underline{\text{sname}}, \underline{\text{pid}}, \text{qty})$.

5.3 Normalization (2)

[**Algorithm 5.3.1**] [**LLJD-BCNF**]

[**Input**] A relation schema R , a set of FDs F in R .

[**Output**] A LLJD D s.t. each new schema in D is in BCNF .

[**Note**] The algorithm may produce different decompositions depending on the order in which FDs are considered.

[**Example 5.3.1**] $\text{Prod\_Manu}(\text{pid}, \text{pname}, \text{price}, \text{mid}, \text{mname}, \text{address})$.

$F = \{(\text{mid} \to \text{Mname}, \text{address}), (\text{pid} \to \text{pname}, \text{mid})\}$, price isn't involved in any functional dependency.

(1) CK: $\{\text{pid}, \text{price}\}$.

Initialization: $D = \{\text{pid pname price mid mname address}\}$.

(2) 1 st iteration:

$R_i(\text{pid}, \text{pname}, \text{price}, \text{mid}, \text{mname}, \text{address})$ is not in BCNF , since $\text{mid} \to \text{mname}$ but mid is not a CK .

Replace R_i by $(\text{mid}, \text{mname}, \text{address})$ and $(\text{pid}, \text{pname}, \text{price}, \text{mid})$.

$D = \{\text{mid mname address}, \text{pid pname price mid}\}$.

$(\underline{\text{mid}}, \text{mname}, \text{address})$ is in BCNF , but $(\underline{\text{pid}}, \text{pname}, \text{price}, \text{mid})$ not.

(2) 2 nd iteration:

$R_i(\text{pid}, \text{pname}, \text{price}, \text{mid})$ is not in BCNF , since pid is not a CK because $\text{pid} \nrightarrow \text{price}$.

Replace R_i by $(\text{pid}, \text{pname}, \text{mid})$ and $(\text{pid}, \text{price})$.

$D = \{\text{mid mname address}, \text{pid pname mid}, \text{pid price}\}$.

$(\text{pid}, \text{pname}, \text{mid})$ is in BCNF since pid is a CK .

$(\text{pid}, \text{price})$ is in BCNF since it has exactly two attributes.

(3) Result: $D = \{\text{mid mname address}, \text{pid pname mid}, \text{pid price}\}$.

[**Definition 5.3.1**] [**Cover**] Let F and G be two sets of FDs in R . F is a **cover** of G if every FD in G can be derived from the FDs in F .

[**Theorem 5.3.1**] Let F and G be two sets of FDs in R . If F is a cover of G and G is a cover of F , then $F=G$.

[**Example 5.3.2**] Show that $F=\{B\to CD, AD\to E, B\to A\}$ is a cover of $G=\{B\to CDE, B\to ABC, AD\to E\}$.

(1) $(AD\to E)$ is in both G and F .

(2) In F , $B\to B$, $B\to CD$, $B\to A$, according to the union rule: $B\to ABCD$.

According to the decomposition rule: $B\to ABC$ which is in G .

(3) In F , $B\to ABCD$, according to the decompostion rule: $B\to AD$.

$AD\to E$, according to the transitivity rule: $B\to E$.

According to the union rule: $B\to ABCDE$.

According to the decomposition rule: $B\to CDE$ which is in G .

In summary, every FD in G can be derived from FDs in F .

[**Definition 5.3.2**] [**Minimal Cover**] Let F be a set of FDs. $F_{\min}$ is a **minimal cover** of F if:

(1) $F_{\min}$ is a cover of F .

(2) Every FD in $F_{\min}$ has single attribute on the right.

(3) No FD in $F_{\min}$ is **redundant**. $(X\to Y)\in F$ is **redundant** if $F-\{X\to Y\}=F$.

(4) For any $(X\to Y)\in F_{\min}$, no attribute in X is **extraneous**. $A\in X$ is **extraneous** if $(X-\{A\})\to Y$ can replace $X\to Y$.

[**Example 5.3.3**] Find $F_{\min}$.

(1) $F=\{A\to BC, B\to C, A\to B, AB\to C\}$.

$F=\{A\to B, A\to C, B\to C, A\to B, AB\to C\}=\{A\to B, B\to C\}=F_{\min}$.

(2) $F=\{A\to AC, B\to ABC, D\to ABC\}$.

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

$$=\{A\to C, B\to A, B\to C, D\to A, D\to B, D\to C\}$$

$$=\{A\to C, B\to A, D\to B\}=F_{\min} .$$

[**Algorithm 5.3.2**] [**LLJD-DPD-3 NF**]

[**Input**] A relation schema R , a set of FDs F in R .

[**Output**] A LLJD and DPD D s.t. each new schema in D is in 3 NF .

[**Note**]

[**Example 5.3.4**] $R(\text{instructor}, \text{cid}, \text{classroom}, \text{text})$ with $F = \{\text{cid} \rightarrow \text{classroom}, \text{text}\}$.

(1) Candidate keys: $\{\text{instructor}, \text{cid}\}$.

(2) $F_{\text{min}} = \{\text{cid} \rightarrow \text{classroom}, \text{cid} \rightarrow \text{text}\}$.

(3) $D = \{\text{cid classroom text}, \text{instructor cid}\}$.

[**Example 5.3.5**] $R = CTHRSG$ with $F = \{C \rightarrow T, CS \rightarrow G, HR \rightarrow C, HS \rightarrow R, HT \rightarrow R\}$.

(1) Candidate keys: $\{H, S\}$.

(2) $F_{\text{min}} = F$.

(3) $D = \{CT, CSG, HRC, HSR, HTR\}$, in which HSR contains the candidate keys.