Database System

北京交通大学软件学院

王方石 教授

E-mail: fshwang@bjtu.edu.cn

Lecture 11

Contents

Chapter 4 Relational Data Theory

4.1 Problems

4.2 Functional Dependency

4.3 Armstrong’s axioms

4.4 The Process of Normalization

4.1 Problems

- **When we design a relational DB, the main objective is to create an accurate representation of the data, its relationships, and constraints.**
- **To achieve this objective, we must identify a suitable set of relations. bad design or good design?**
- **Normalization in this chapter is the technique that we can use to help identify such suitable relations.**

Data Redundancy

- **Major aim of GOOD relational database design is to group attributes into relations to minimize data redundancy and reduce file storage space required by base relations.**

- **Problems associated with data redundancy are illustrated by comparing the following Staff and Branch relations with the StaffBranch relation.**

Data Redundancy

Staff

staffNo	sName	position	salary	branchNo
SL21	John White	Manager	30000	B005
SG37	Ann Beech	Assistant	12000	B003
SG14	David Ford	Supervisor	18000	B003
SA9	Mary Howe	Assistant	9000	B007
SG5	Susan Brand	Manager	24000	B003
SL41	Julie Lee	Assistant	9000	B005

Branch

branchNo	bAddress
B005	22 Deer Rd, London
B007	16 Argyll St, Aberdeen
B003	163 Main St, Glasgow

StaffBranch

staffNo	sName	position	salary	branchNo	bAddress
SL21	John White	Manager	30000	B005	22 Deer Rd, London
SG37	Ann Beech	Assistant	12000	B003	163 Main St, Glasgow
SG14	David Ford	Supervisor	18000	B003	163 Main St, Glasgow
SA9	Mary Howe	Assistant	9000	B007	16 Argyll St, Aberdeen
SG5	Susan Brand	Manager	24000	B003	163 Main St, Glasgow
SL41	Julie Lee	Assistant	9000	B005	22 Deer Rd, London

Update Anomalies（更新异常）

- Relations that contain redundant information may potentially suffer from update anomalies.

- Types of update anomalies include:
  - Insertion
  - Deletion
  - Modification

Example sct（sno, cno, tno, sname, grade, cname）

S(sno, sname, age, sex)

C(cno,cname,tno)

SC(sno,cno,grade)

update anomalies

- Insertion,
- Deletion
- Modification

SCT

Sno	Cno	Tno	Sname	Grade	Cname
S1	C1	T1	赵民	90	OS
S1	C2	T2	赵民	90	DS
S1	C3	T3	赵民	85	C++
S1	C4	T4	赵民	87	DB
S2	C1	T4	李军	90	OS
S3	C1	T4	陈江	75	OS
S3	C2	T2	陈江	70	DS
S3	C4	T4	陈江	56	DB
S4	C1	T1	魏致	90	OS
S4	C2	T2	魏致	85	DS
S5	C1	T1	乔远	95	OS
S5	C4	T4	乔远	80	DB

Reason: Dependency among data is too strong.

4.2 Functional Dependency

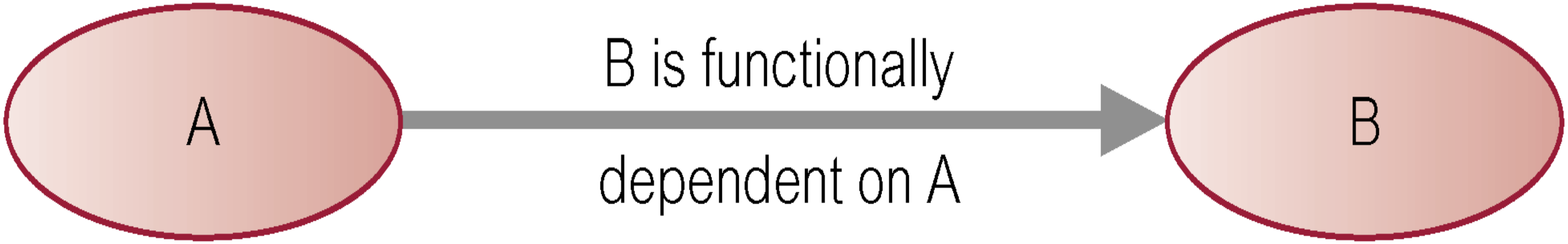
- **Main concept associated with normalization.**
- **Functional Dependency**
  - **Describes relationship between attributes in a relation.**
  - **The Functional Dependency is specified as a constraint between the attributes.**
  - **If A and B are attributes of relation R, B is functionally dependent on A (denoted $A \rightarrow B$), if each value of A in R is associated with exactly one value of B in R.**

Strict Definition of Functional Dependencies

- $X \rightarrow Y$ is an assertion about a relation R that whenever two tuples of R agree on all the attributes of X, then they must also agree on all attributes in set Y.
  - Say "X ->Y holds in R."
  - Convention: …, X, Y, Z represent sets of attributes; A, B, C,… represent single attributes.
  - Convention: no set formers { } in sets of attributes, just ABC, rather than $\{A,B,C\}$.

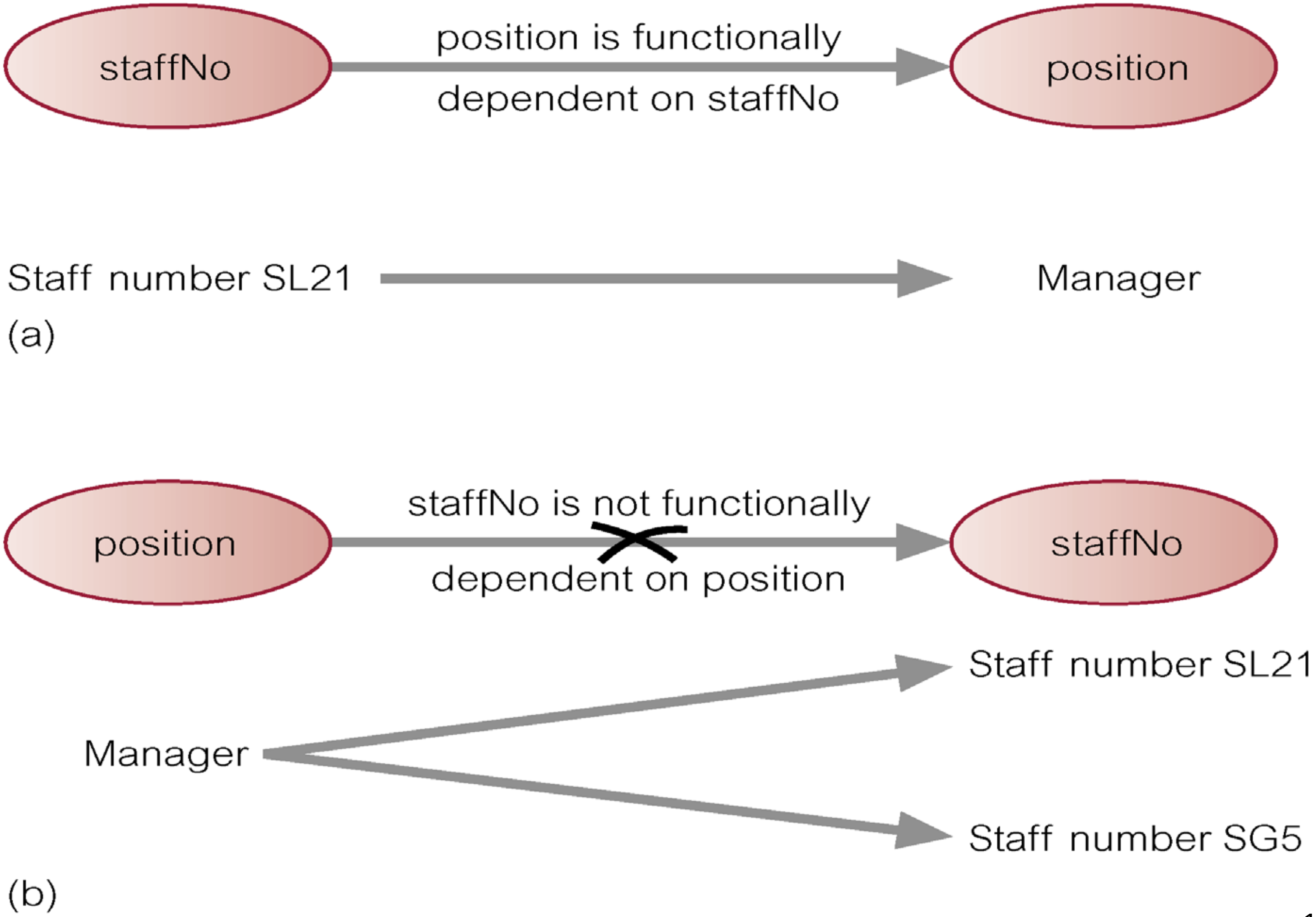
Functional Dependency (FD)

- **FD is a property of the meaning or semantics of the attributes in a relation.**
- **The semantics indicate how attributes relate to one another. Diagrammatic representation:**



Determinant (决定因素) of a functional dependency refers to attribute or group of attributes on left-hand side of the arrow.

Example - Functional Dependency



Functional Dependency

- **Main characteristics of functional dependencies used in normalization:**
  - **have a 1:1 relationship between attribute(s) on left and right-hand side of a dependency;**
  - **hold for all time;**
  - **are nontrivial.** （$X \to Y$ **is trivial if** $Y \subseteq X$.）

S(sno,sname,age,sex)

$sno \to sname$? **Yes!**

$sno \to$ age ? **Yes!**

sname → sno? **No!** **It dose not hold for all time.**

How to determine the candidate keys

★（1）Given FD set， How to determine the candidate keys (CK)?

1) If an attribute does not appear in FD's, then it must be included in CK.

2) If an attribute does not appear on the right side of any FD, then it must be included in CK.

3) If the attribute or attribute set can identify one tuple uniquely , then it must be a CK.

Example:

1. $R(A,B,C,D)$，$F=\{A\rightarrow D, B\rightarrow C\}$

   CK= (A,B)

2. $R(A,B,C,D)$，$F=\{A\rightarrow D, D\rightarrow C\}$

   CK= (A,B)

3. $R(A,B,C,D)$，$F=\{A\rightarrow C, B\rightarrow C\}$

   CK= (A,B,D)

★（2）If FD’s is not given, How to determine the candidate keys (CK)?

We can determine the candidate keys based on the semantic description of DB, at the same time we also can give the FD’s on the relation schema.

★If the entire tuple is CK, it is call **Full-Key**.

e.g. R(P,W,A)， P is the player number.

W is the work number. A is the audient number.

Its CK is (P,W,A), Full-key.

Example: **R1(zip, city, street)**

F1= {zip→city, (city, street)→zip}

CK1= (city, street) **CK2= (zip, street)**

R2 (sno,sname, cno, grade)

F2={sno→sname, sname →sno, (sno,cno) →grade}

CK1= (sno, cno) **CK2= (sname, cno)**

F2’={sno→sname, (sno,cno) →grade}

CK1= (sno, cno)

Chapter 4 Relational Data Theory

4.1 Problems

4.2 Functional Dependency

4.3 Armstrong's axioms

4.4 The Process of Normalization

Inferring Functional Dependency

- We are given FD's $X_1 \text{ -> } A_1, X_2 \text{ -> } A_2, \ldots, X_n \text{ -> } A_n$, and we want to know whether an FD $Y \text{ -> } B$ must hold in any relation that satisfies the given FD's.
  – Example: If $A \text{ -> } B$ and $B \text{ -> } C$ hold, surely $A \text{ -> } C$ holds, even if we don't say so.
- Important for design of good relation schemas.

- **The set of inference rules, called Armstrong's axioms, specifies how new functional dependencies can be inferred from the given ones.**

4.3 Armstrong's axioms

- **Let A, B, and C be subsets of the attributes of relation R. Armstrong's axioms are as follows:**

  1. Reflexivity

  If B is a subset of A, then $A \rightarrow B$

  2. Augmentation

  If $A \rightarrow B$, then $A \cup C \rightarrow B \cup C$

  3. Transitivity

  If $A \rightarrow B$ and $B \rightarrow C$, then $A \rightarrow C$

We use axioms to normalize relations and remove the three kinds of anomalies.

Splitting rule

Splitting Right Sides of FD's

- $X \rightarrow A_1A_2...A_n$ holds for R exactly when each of $X \rightarrow A_1$, $X \rightarrow A_2$,..., $X \rightarrow A_n$ hold for R.
- Example: $A \rightarrow BC$ is equivalent to

  $A \rightarrow B$ and $A \rightarrow C$.
- There is no splitting rule for left sides.

  e.g. XY→A without X→A and(or) Y→A
- We'll generally express FD's with singleton right sides.

Closure of F

The set of all functional dependencies implied by a given set of the functional dependencies F called closure of F (written F^+).

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

e.g. R=ABC, F={**A→B, B→C**}, **calculate** F^+

F^+ ={ Φ→Φ,

A→Φ,	B→Φ,	C→Φ,	AB→Φ,	AC→Φ,	BC→Φ,	ABC→Φ,
A→A,	B→B,	C→C,	AB→A,	AC→A,	BC→B,	ABC→A,
A→B,	**B→C,**		AB→B,	AC→B,	BC→C,	ABC→B,
A→C,	B→BC,		AB→C,	AC→C,	BC→BC,	ABC→C,
A→AB,			AB→AB,	AC→AB,		ABC→AB,
A→AC,			AB→AC,	AC→AC,		ABC→AC,
A→BC,			AB→BC,	AC→BC,		ABC→BC,
A→ABC,			AB→ABC,	AC→ABC,		ABC→ABC

} 43 FD in total， **F≡ F^+**

显然，{A → B， B → C}就是R上的最小函数依赖集。

Equivalence of Functional Dependency Set

- The Closure of functional dependencies for a given relation can be very large.
- It is important to find an approach that can reduce set to a manageable size.
- Two functional dependencies set, F1 and F2, say F1 is equivalent to F2

$$F1 \equiv F2$$

if and only if

Every FD in F1 can be inferred from F2, and every FD in F2 can be inferred from F1.

Example 1

$F1=\{A\rightarrow BC, B\rightarrow C\}$, $F2=\{A\rightarrow B, B\rightarrow C\}$

$F1\equiv F2$? Yes!

(1) $F1 \Rightarrow F2$

According to **Splitting** rule, we have

$F1=\{A\rightarrow B, A\rightarrow C, B\rightarrow C\}$

(2) $F2 \Rightarrow F1$

$B\rightarrow C \Rightarrow B\rightarrow BC$ (Augmentation)

$A\rightarrow B\rightarrow BC$ (Transitivity)

Example 2

F1={AB→C,B→C}, F2={B→C}

F1≡F2? Yes!

(1) F1 ⇒ F2

Obviously

(2) F2 ⇒ F1

B→C ⇒ AB →AC (Augmentation)

AC→C (Reflexivity)

AB →AC →**C**(Transitivity)

Equivalent FD set

(1) Sets of functional dependencies may have **redundant dependencies** that can be inferred from the others.

– e.g: $A \rightarrow C$ is redundant in: $\{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow C\}$

(2) **Parts** of a functional dependency may be **redundant**.

e.g. on **Right-hand** Side: $\{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow \mathbf{CD}\}$

can be simplified to $\{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow D\}$

1) $\because \mathbf{A \rightarrow CD} \quad \therefore \mathbf{A \rightarrow C}$ **and** $\mathbf{A \rightarrow D}$

2) $\because A \rightarrow B, \ B \rightarrow C \therefore A \rightarrow C$ => $A \rightarrow AC$ (**Augmentation**)

$\because A \rightarrow D \therefore AC \rightarrow CD \therefore A \rightarrow AC \rightarrow CD$

e.g. on **Left-hand** Side: $\{A \rightarrow B, \ B \rightarrow C, \ \mathbf{AC} \rightarrow D\}$

can be simplified to $\{A \rightarrow B, \ B \rightarrow C, \ A \rightarrow D\}$

1) $\mathbf{A \rightarrow B, \ B \rightarrow C}$ => $\mathbf{A \rightarrow C}$ => $\mathbf{A \rightarrow AC \rightarrow D}$ => $\mathbf{A \rightarrow D}$

2) $\mathbf{A \rightarrow D}$ => $\mathbf{AC \rightarrow CD \rightarrow D}$ => $\mathbf{AC \rightarrow D}$

Canonical Cover（正则覆盖）

A *canonical cover* for *F* is a set of dependencies $F_{canonical}$ such that

- *F* logically implies all dependencies in $F_{canonical}$, and $F_{canonical}$ logically implies all dependencies in *F*

  i.e. $F_{canonical} \equiv F$

- All functional dependency in $F_{canonical}$ have **singleton right side**, and

- No attribute of **left-hand** side of each functional dependency in F_c is extraneous, and

- No functional dependency in $F_{canonical}$ is extraneous,

Canonical Cover

- **Intuitively, a canonical cover of F is a "minimal" set of functional dependencies （最小函数依赖集）equivalent to F,** having no redundant dependencies or redundant parts in dependencies.

- **A canonical cover is not unique** for a given set of functional dependencies, therefore one set F can have multiple covers $F_{canonical}$

Computing a Canonical Cover

The ***order*** of the following three steps ***is not fixed***. Different orders can get different results.

Consider a set *F* of functional dependencies.

(1) For each functional dependency **X→Y** in *F,*

make Y include a **single attribute**.

It guarantees any attribute on right-hand side (Y) is not extraneous.

e.g. AB →**CD** ⇨ { AB →**C** , AB →**D** }

Computing a Canonical Cover

(2) For each functional dependency $X \rightarrow Y$ in *F*,

check if $F \equiv F - \{X \rightarrow Y\}$ holds.

i.e. whether you can infer $X \rightarrow Y$ using only FDs

in $F - \{X \rightarrow Y\}$? **$X \rightarrow Y$ is extraneous?**

if it does, $X \rightarrow Y$ is **extraneous** and remove it from F.

e.g. $F = \{A \rightarrow B,\ \ B \rightarrow C,\ \ A \rightarrow C\}$

$\because A \rightarrow B \rightarrow C \qquad \therefore A \rightarrow C$

so **$A \rightarrow C$ is extraneous**

$F \equiv F - \{A \rightarrow C\}$

Computing a Canonical Cover (cont)

(3) For each functional dependency $X \rightarrow Y$ ($A \subset X$) in *F*,

check if $F \equiv (F-\{X \rightarrow Y\}) \cup \{A \rightarrow Y\}$ holds.

i.e. whether you can infer $X \rightarrow Y$ using FDs

in $(F-\{X \rightarrow Y\}) \cup \{A \rightarrow Y\}$?

Any attribute on left-hand side (X) is extraneous?

if it does, (X-A) is **extraneous** attribute(s)

and remove $X \rightarrow Y$ and add $A \rightarrow Y$ in F.

e.g. **F= {A → B, B → C, AC → D}**

$\because A \rightarrow B \rightarrow C \quad \therefore A \rightarrow C \therefore A \rightarrow AC \rightarrow D$

$\because A \rightarrow D \therefore AC \rightarrow DC \rightarrow D \therefore AC \rightarrow D$

$F \equiv (F-\{AC \rightarrow D\}) \cup \{A \rightarrow D\} = \{A \rightarrow B, B \rightarrow C, A \rightarrow D\}$

Example of Computing a Canonical Cover

Example1： $R = (A, B, C)$

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

(1) decompose $A \rightarrow BC$ to $A \rightarrow B$ and $A \rightarrow C$

F is now $\{A \rightarrow B, A \rightarrow C, B \rightarrow C, AB \rightarrow C\}$

(2) $\because A \rightarrow C$ *and* $B \rightarrow C \therefore AB \rightarrow C$

then Remove $AB \rightarrow C$

F is now $\{A \rightarrow B, A \rightarrow C, B \rightarrow C\}$

(3) $\because A \rightarrow$ B *and* $B \rightarrow C \therefore A \rightarrow C$

then Remove $A \rightarrow C$

F is now $\{A \rightarrow B, B \rightarrow C\}$

The canonical cover is: $\{ A \rightarrow B, B \rightarrow C \}$

Example

Example:2: $\mathbf{F = \{A \rightarrow B,\ \ B \rightarrow C,\ \ A \rightarrow C\}}$

$\because A \rightarrow B \rightarrow C \ \ \therefore A \rightarrow C,$

so **$A \rightarrow C$ is extraneous**

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

Example 3: $\mathbf{F = \{A \rightarrow B,\ \ B \rightarrow C,\ \ A \rightarrow CD\}}$

$\because A \rightarrow \mathbf{CD} \ \ \therefore A \rightarrow C,\ \ A \rightarrow D$

$\because A \rightarrow B \rightarrow C \ \ \therefore A \rightarrow C\ ,$

so **$A \rightarrow C$ is extraneous**

$F_c = \{A \rightarrow B,\ \ B \rightarrow C,\ \ A \rightarrow D\}$

4.4 The Process of Normalization

4.4.1 Definition of normal form

4.4.2 Definition of normalization

4.4.3 Standards for schema decomposition

4.4.4 UNF &1NF & 2NF

4.4.5 3NF & BCNF

4.4 The Process of Normalization

- **Normalization is a formal technique （形式化技术）for analyzing a relation based on its primary key and functional dependencies between its attributes.**

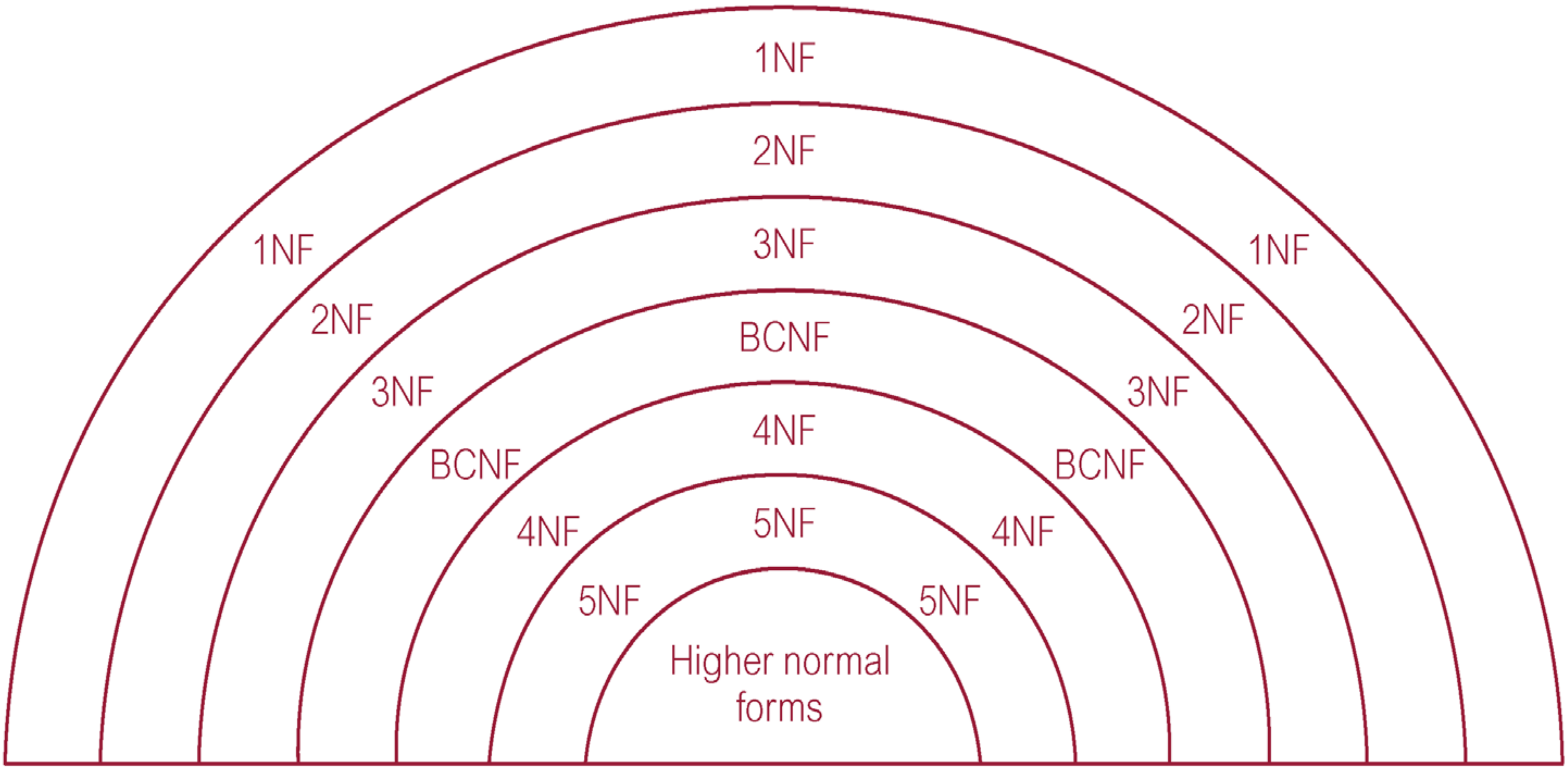
- **It is often executed as a series of steps. Each step corresponds to a specific normal form that has known properties.**

- **As normalization proceeds, relations become progressively more restricted (stronger) in format and also less vulnerable to update anomalies.**

4.4.1 Definition of Normal Form

- **Normal form is the set of relation schema that conform to some specific requirements.**
- **Different level of normal forms conforms to different degree of requirements.**
- **Four most commonly used normal forms are first (1NF), second (2NF) and third (3NF) normal forms, and Boyce–Codd normal form (BCNF).**
- **All these normal forms are based on functional dependencies among the attributes of a relation.**
- **A relational schema can be normalized to a specific form to prevent the possible occurrence of update anomalies（更新异常）.**

Relationship Between Normal Forms



1NF ⊃ 2NF ⊃ 3NF ⊃ BCNF ⊃ 4NF ⊃ 5NF

4.4.2 Definition of Normalization

A relation schema in a lower level of normal form is transformed to a set of several relation schemas in a higher level of normal form through schema decomposition. This process is called ***normalization***.

Simply put, schema decomposition is to distribute the data in one table into several tables.

4.4.3 Standards for schema decomposition

- When we decompose a relation schema R with a set of functional dependencies F into R_1, R_2,.., R_n we want
  - **No redundancy**: The relations R_i preferably should be in either Boyce-Codd Normal Form or Third Normal Form.
  - **Lossless-join decomposition**: Otherwise decomposition would result in information loss.
  - **Dependency preservation**: Let F_i be the set of dependencies F^+ that include only attributes in R_i.
    - Preferably the decomposition should be dependency preserving, that is,

$$(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$$

Two important properties of decomposition

1. ***Lossless-join property*** **enables us to find any instance of original relation from corresponding instances in the smaller relations.**

   In one word, Information before and after decomposition are the same.

Example

$R(A, B, C)$, $\rho = \{R_1, R_2\}$ $R_1 = (AB), R_2 = (BC)$,
$r_1 = \pi_{R1}(r)$, $r_2 = \pi_{R2}(r)$, 求r_1, r_2, $m_\rho(r)$

r

A	B	C
a1	b1	c1
a2	b1	c2
a1	b1	c2

r1

A	B
a1	b1
a2	b1

r2

B	C
b1	c1
b1	c2

$m_\rho(r)$

A	B	C
a1	b1	c1
a1	b1	c2
a2	b1	c1
a2	b1	c2

r<>$m_\rho(r)$

So Lossy-Join Decomposition

Two important properties of decomposition

2. *Dependency preservation property* **enables us to enforce a constraint on original relation by enforcing some constraint on each of the smaller relations.**

In one word, the sets of functional dependency keep equivalent before and after decomposition.

Example $R(sno,dept,director)$, $\rho=\{R_1,R_2\}$

$F= = \{sno \rightarrow dept, \quad dept \rightarrow director\}$

$R_1=(sno,dept), R_2=(sno,director)$,

$r_1=\pi_{R1}(r), r_2=\pi_{R2}(r)$, 求$r_1$, r_2, $m_\rho(r)$

r

sno	dept	director
s1	d1	John
s2	d1	John
s3	d2	Kate
s4	d3	Jane

r_1

sno	dept
s1	d1
s2	d1
s3	d2
s4	d3

r_2

sno	director
s1	John
s2	John
s3	Kate
s4	Jane

$m_\rho(r)$

sno	dept	director
s1	d1	John
s2	d1	John
s3	d2	Kate
s4	d3	Jane

It is Lossless-Join Decomposition, but it misses $dept \rightarrow director$ **.**

No Dependency preservation

How to check if Decomposition is lossless?

- All attributes of an original schema (*R*) must appear in the decomposition (R_1, R_2):

$$R = R_1 \cup R_2$$

- A decomposition of R into R_1 and R_2 is lossless join if and only if at least one of the following dependencies is in F^+:
  - $R_1 \cap R_2 \rightarrow R_1 - R_2$
  - $R_1 \cap R_2 \rightarrow R_2 - R_1$

Example

$R = (A, B, C)$, $F = \{A \rightarrow B, B \rightarrow C)$

Can be decomposed in 3 different ways

(1) $R_1 = (A, B)$, $R_2 = (B, C)$

– Lossless-join decomposition:

$R_1 \cap R_2 = \{B\}$, R_2 - $R_1 = \{C\}$ and $B \rightarrow C$

– Dependency preserving

Both $A \rightarrow B$ *and* $B \rightarrow C$ hold.

(2) $R_1 = (A, B)$, $R_2 = (A, C)$

– Lossless-join decomposition:

$R_1 \cap R_2 = \{A\}$, R_1 - $R_2 = \{B\}$ and $A \rightarrow B$

– Not dependency preserving

$A \rightarrow B$ *hold, but* $B \rightarrow C$ does not hold

Example

$R = (A, B, C)$, $F = \{A \rightarrow B, B \rightarrow C)$

(3) $R_1 = (A, C)$, $R_2 = (B, C)$

– Lossy-join decomposition:

$R_1 \cap R_2 = \{C\}$, $R_1 - R_2 = \{A\}$ and $R_2 - R_1 = \{B\}$

But neither $C \rightarrow$A nor $C \rightarrow$B holds.

– No Dependency preserving

$B \rightarrow C$ holds. *But* $A \rightarrow B$ dose not hold.

Algorithm of Checking Lossless-Join

Input： R(U,F, ρ)

$U = A_1 A_2 A_3 \dots A_n$,

FD F,

decomposition $\rho=\{R_1, R_2, \dots R_k\}$

output :

return True if ρ keep Lossless-Join property,

otherwise return False.

```
Check-Lossless (R, F, ρ)
{  Construct the initial table Rρ ;
   for (each FD X→Y in F)
       { if(ti1[X]=ti2[X]=…=tim [X] in Rρ)
           {make ti1[Y],ti2[Y],…,tim[Y] get the same value}
         if (there is a line like a1,a2,…an in Rρ)
             {return True ;  }
       }
   if (there is a line like a1,a2,…an in Rρ )
         {return True;  }   else {return false ;  }
}
```

Example: Given R(U,F) ,

$U=\{SNO,CNO,GRADE,TNAME,TAGE,OFFICE\}$

$F=\{(SNO,CNO)\rightarrow GRADE, CNO\rightarrow TNAME,$

$TNAME\rightarrow(TAGE,OFFICE)\}$

There are two decompositions as follows.

$\rho_1=\{SC, CT, TO\}$, $\rho_2=\{SC, GTO\}$

where $SC=\{SNO,CNO,GRADE\}$, $CT=\{CNO,TNAME\}$,

$TO=\{TNAME,TAGE,OFFICE\}$

$GTO=\{GRADE,TNAME,TAGE,OFFICE\}$

Check if ρ_1,ρ_2 keep Lossless-Join property.

solution: ρ_1 is , ρ_2 is not.

U={SNO,CNO,GRADE,TNAME,TAGE,OFFICE}

SC={SNO,CNO,GRADE},CT={CNO,TNAME},

TO={TNAME,TAGE,OFFICE}

GTO={GRADE,TNAME,TAGE,OFFICE}

ρ_1={SC, CT, TO}

	SNO	CNO	GRADE	TNAME	TAGE	OFFICE
SC	a_1	a_2	a_3	b_{14}	b_{15}	b_{16}
CT	b_{21}	a_2	b_{23}	a_4	b_{25}	b_{26}
TO	b_{31}	b_{32}	b_{33}	a_4	a_5	a_6

F={ (SNO,CNO)→GRADE,

CNO→TNAME,

TNAME→(TAGE,OFFICE)}

	SNO	CNO	GRADE	TNAME	TAGE	OFFICE
SC	a_1	a_2	a_3	a_4	a_5	a_6
CT	b_{21}	a_2	b_{23}	a_4	a_5	a_6
TO	b_{31}	b_{32}	b_{33}	a_4	a_5	a_6

ρ1 keeps Lossless-Join property.

U={SNO,CNO,GRADE,TNAME,TAGE,OFFICE}

SC={SNO,CNO,GRADE},CT={CNO,TNAME},

GTO={GRADE,TNAME,TAGE,OFFICE}

ρ_2={SC,GTO}

	SNO	CNO	GRADE	TNAME	TAGE	OFFICE
SC	a_1	a_2	a_3	b_{14}	b_{15}	b_{16}
GTO	b_{21}	b_{22}	a_3	a_4	a_5	a_6

$F=\{ (SNO,CNO)\rightarrow GRADE,$

$CNO\rightarrow TNAME,$

$TNAME\rightarrow(TAGE,OFFICE)\}$

	SNO	CNO	GRADE	TNAME	TAGE	OFFICE
SC	a_1	a_2	a_3	b_{14}	b_{15}	b_{16}
GTO	b_{21}	b_{22}	a_3	a_4	a_5	a_6

ρ_2 does not keep Lossless-Join property.

4.4.4 UNF &1NF & 2NF

（1） Unnormalized Form (UNF)

A table that contains non-atomic elements.

eno	name	salary		
		basic	bonus	tax
e1	John	$900.00	$120	$45
e2	Lily	$1200.00	$180	$96

（2） First Normal Form (1NF)

- Domain is atomic if its elements are considered to be indivisible units
- A relational schema R is in first normal form if the domains of all attributes of R are atomic
- **A relation in which intersection of each row and column contains one and only one value.**

UNF to 1NF

Separate non-atomic elements into individual columns.

salary

eno	name	basic	bonus	tax
e1	John	$900.00	$120	$45
e2	Lily	$1200.00	$180	$96

A relation schema in 1NF must not be good.

student

sno	cno	sname	cname	grade	sex	credit
s1	c1	ss1	OS	79	f	4
s1	c2	ss1	DB	90	f	3
s2	c1	ss2	OS	85	m	4

（3）Second Normal Form (2NF)

- **Based on concept of full functional dependency:**
  - **A and B are attributes of a relation,**
  - **B is fully dependent on A if B is functionally dependent on A but not on any proper subset of A.**

  If $A \rightarrow B$, and for any $X \subset A$, $X \not\rightarrow B$, then $A \xrightarrow{f} B$

- **2NF - A relation that is in 1NF and every non-primary-key attribute (not a part of any candidate-key)is fully functionally dependent on any candidate key.**

Example 1

R(sno,cno,sname,cname,sex,grade,credit)，

R∈2NF？

sno	cno	sname	cname	grade	sex	credit
s1	c1	ss1	OS	79	f	4
s1	c2	ss1	DB	90	f	3
s2	c1	ss2	OS	85	m	4

No! F={sno→sname,cno→cname,sno →sex, (sno,cno) →grade, cno →credit}

∵ CK=(sno,cno) and (sno,cno) $\xrightarrow{P}$ sex

non-primary-key attribute

∴ R∈1NF

Dose there exist three kinds of anomalies?

Decompose 1NF to 2NF

- **Identify candidate key for the 1NF relation.**
- **Identify functional dependencies in the relation.**
- **If partial dependencies exist on any candidate key, remove them by placing them in a new relation along with copy of their determinant.**

Example 2

R(sno,sname, cno, grade)

F={sno→sname, (sno,cno) →grade}

R∈2NF?

No! ∵ there is a non-candidate-key attribute ***sname*** which is partially dependent on the candidate key **(sno,cno)**. ∴ R∈1NF

[decompose]

R1=(sno, sname) R2=(sno,cno, grade)

R1 & R2∈2NF

Example 3

R(sno,sname, dept,buildingNo)

F={sno→sname, sno →dept, dept→buildingNo}

R∈2NF?

sno	sname	dept	buildingNo
s1	ss1	CS	8
s2	ss2	EE	9
s3	ss3	EE	9

Yes! ∵ there is no non-primary-key attribute which is partially functionally dependent on any candidate key. ∴ R∈2NF

Dose there exist three kinds of anomalies?

4.4.5 3NF & BCNF

（1）Third Normal Form (3NF)

- **Based on concept of transitive dependency:**
  - **A, B and C are attributes of a relation such that if A → B and B → C,**
  - **then C is transitively dependent on A through B. (Provided that A is not functionally dependent on B or C).**

- **3NF - A relation that is in 1NF and 2NF and in which no non-primary-key attribute is transitively dependent on any candidate key.**

Example 4

R(sno,sname, dept,buildingNo)

F={sno→sname, sno →dept, dept→buildingNo}

R∈3NF?

sno	sname	dept	buildingNo
s1	ss1	CS	8
s2	ss2	EE	9
s3	ss3	EE	9

No! ∵ there is a non-primary-key attribute buildingNo which is transitively dependent on the candidate key sno. ∴ R ∉ 3NF

Dose there exist three kinds of anomalies? **Yes!**

Decompose 2NF to 3NF

- **Identify the primary key in the 2NF relation.**
- **Identify functional dependencies in the relation.**
- **If transitive dependencies exist on the primary key，remove them by placing them in a new relation along with copy of their determinant.**

Example 5

R(sno,sname, dept, buildingNo)

F={sno→sname, sno →dept, dept→buildingNo}

R1(sno,sname, dept) , R2(dept,buildingNo)

R1∈3NF？ R2∈3NF？

Yes! ∵ there is a no non-primary-key attribute which is partially or transitively dependent on the candidate key.

∴ R1&R2∈3NF

Dose there exist three kinds of anomalies? No!

Example 6

How about R(sno,sname, cno, grade) and

F={sno→sname, sname →sno, (sno,cno) →grade}?

R∈3NF?

Yes! ∵ there is a no non-candidate-key attribute which is partially or transitively dependent on the candidate key. ∴ R∈3NF

Dose there exist three kinds of anomalies? Yes!

sno	sname	cno	grade
s1	ss1	C1	81
s1	ss1	C2	92
s2	ss2	C1	79

（2）Boyce–Codd Normal Form (BCNF)

- **Based on functional dependencies that take into account all candidate keys in a relation, however BCNF also has additional constraints compared with general definition of 3NF.**

- **BCNF - A relation is in BCNF if and only if every determinant is a candidate key.**

Boyce–Codd normal form (BCNF)

- **Difference between 3NF and BCNF is that for a functional dependency *A* → *B*, 3NF allows this dependency in a relation if *B* is a primary-key attribute and *A* is not a candidate key.** e.g. R(sno,sname, cno, grade)

  sno→sname, sname →sno, (sno,cno) →grade

- **Whereas, BCNF insists that for this dependency to remain in a relation, *A* must be a candidate key.**

- **Every relation in BCNF is also in 3NF. However, relation in 3NF may not be in BCNF.**

Boyce–Codd normal form (BCNF)

- **Violation of BCNF in 3NF is quite rare.**
  （3NF中不是BCNF的很少）
- **Potential to violate BCNF may occur in a relation that:**
  - **contains two (or more) composite candidate keys （A candidate key that consists of two or more attributes.）;**
  - **the candidate keys overlap (i.e. have at least one attribute in common).**

Example 7

R(sno,sname, cno, grade) and

F={sno→sname, sname →sno, (sno,cno) →grade}

R1(sno,sname) , R2(sno, cno, grade)

F1={sno→sname, sname→sno}, F2={ (sno,cno)→grade}

R1∈BCNF? R2∈BCNF?

Yes! ∵ Every determinant is a candidate key

∴ R1&R2∈BCNF

Dose there exist three kinds of anomalies? **No!**

Example 8: R1(zip, city, street)

F1= {zip→city, (city, street)→zip}

CK1= (city, street) CK2= (zip, street)

two overlap and **composite candidate keys**

R1 ∉ **BCNF**

Example 9: R2 (S,P,J)

where S is the student number, J is course number and P is the grade rank in one course. Given one student and one course, there is unique rank to correspond to him or her. Given one course and one rank, , there is unique student to correspond to this position.

F1= { $SJ \rightarrow P$, $JP \rightarrow S$ }

CK1= (S, J) **CK2= (J, P)**

two overlap and **composite candidate keys**

R1$\in$BCNF

Example10: R3 (S,T,J)

where S is the student number, J is course number and T is the teacher number. Given one student and one course, there is unique teacher to correspond to him or her. Each teacher teaches only one course.

F1= { $T \rightarrow J$, $SJ \rightarrow T$ }

CK1= (S, T) **CK2= (S, J)**

two overlap and **composite candidate keys**

R1 $\notin$ **BCNF**

Conclusion: process of normalization

1NF

Eliminate the non-primary attribute's partially functional dependency on CK

2NF

Eliminate the non-primary attribute's partially and transitively functional dependency on CK.

3NF

Eliminate the primary partially and transitively functional dependency on CK.

BCNF