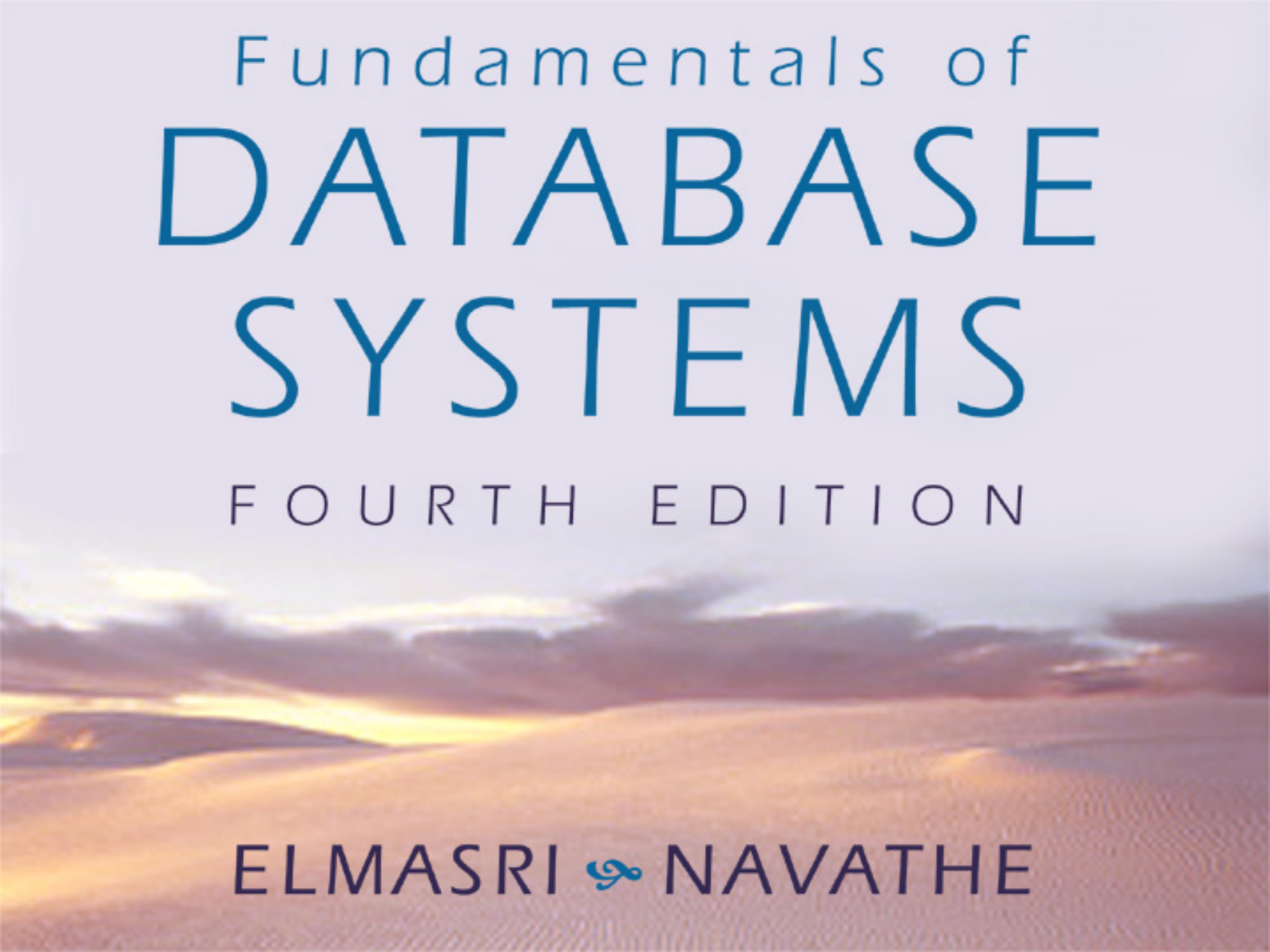
Fundamentals of DATABASE SYSTEMS

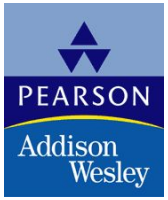
FOURTH EDITION



ELMASRI NAVATHE

Chapter 10

Functional Dependencies and Normalization for Relational Databases



Chapter Outline

1 Informal Design Guidelines for Relational Databases
- 1.1Semantics of the Relation Attributes
- 1.2 Redundant Information in Tuples and Update Anomalies
- 1.3 Null Values in Tuples
- 1.4 Spurious Tuples

2 Functional Dependencies (FDs)
- 2.1 Definition of FD
- 2.2 Inference Rules for FDs
- 2.3 Equivalence of Sets of FDs
- 2.4 Minimal Sets of FDs

Chapter Outline(contd.)

3 Normal Forms Based on Primary Keys

- 3.1 Normalization of Relations
- 3.2 Practical Use of Normal Forms
- 3.3 Definitions of Keys and Attributes Participating in Keys
- 3.4 First Normal Form
- 3.5 Second Normal Form
- 3.6 Third Normal Form

4 General Normal Form Definitions (For Multiple Keys)

5 BCNF (Boyce-Codd Normal Form)

1 Informal Design Guidelines for Relational Databases (1)

- What is relational database design?

  The grouping of attributes to form "good" relation schemas
- Two levels of relation schemas
  - The logical "user view" level
  - The storage "base relation" level
- Design is concerned mainly with base relations
- What are the criteria for "good" base relations?

Informal Design Guidelines for Relational Databases (2)

- We first discuss informal guidelines for good relational design
- Then we discuss formal concepts of functional dependencies and normal forms
  - 1NF (First Normal Form)
  - 2NF (Second Normal Form)
  - 3NF (Third Normal Form)
  - BCNF (Boyce-Codd Normal Form)
- Additional types of dependencies, further normal forms, relational design algorithms by synthesis are discussed in Chapter 11

1.1 Semantics of the Relation Attributes

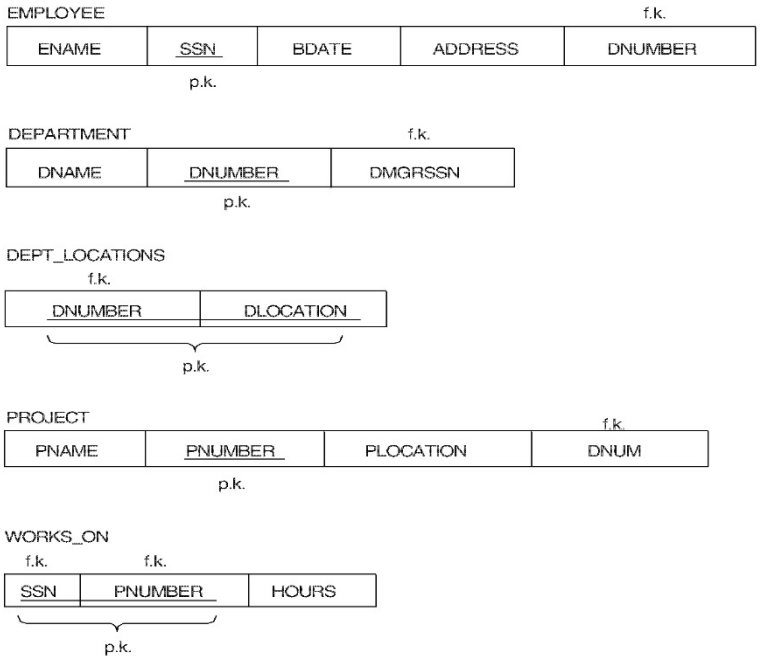
GUIDELINE 1: Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).

- Attributes of different entities (EMPLOYEEs, DEPARTMENTs, PROJECTs) should not be mixed in the same relation
- Only foreign keys should be used to refer to other entities
- Entity and relationship attributes should be kept apart as much as possible.

Bottom Line: Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.

Figure 10.1 A simplified COMPANY relational database schema

Figure 14.1 Simplified version of the COMPANY relational database schema.



Note: The above figure is now called Figure 10.1 in Edition 4

1.2 Redundant Information in Tuples and Update Anomalies

- Mixing attributes of multiple entities may cause problems
- Information is stored redundantly wasting storage
- Problems with update anomalies
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

EXAMPLE OF AN UPDATE ANOMALY (1)

Consider the relation:

EMP_PROJ (Emp#, Proj#, Ename, Pname, No_hours)

- **Update Anomaly:** Changing the name of project number P1 from "Billing" to "Customer-Accounting" may cause this update to be made for all 100 employees working on project P1.

EXAMPLE OF AN UPDATE ANOMALY (2)

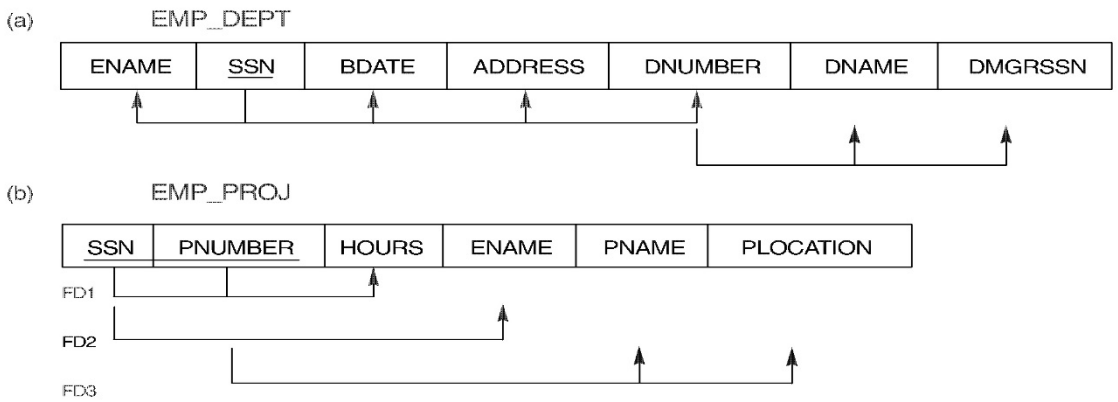
- **Insert Anomaly:** Cannot insert a project unless an employee is assigned to .

  Inversely - Cannot insert an employee unless an he/she is assigned to a project.

- **Delete Anomaly:** When a project is deleted, it will result in deleting all the employees who work on that project. Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

Figure 10.3 Two relation schemas suffering from update anomalies

Figure 14.3 Two relation schemas and their functional dependencies. Both suffer from update anomalies. (a) The EMP_DEPT relation schema. (b) The EMP_PROJ relation schema.



Note: The above figure is now called Figure 10.3 in Edition 4

Figure 10.4 Example States for EMP_DEPT and EMP_PROJ

Figure 14.4 Example relations for the schemas in Figure 14.3 that result from applying NATURAL JOIN to the relations in Figure 14.2. These may be stored as base relations for performance reasons.

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	DNAME	DMGRSSN
Smith,John B.	123456789	1965-01-09	731 Fondren,Houston,TX	5	Research	333445555
Wong,Franklin T.	333445555	1955-12-08	638 Voss,Houston,TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle,Spring,TX	4	Administration	987654321
Wallace,Jennifer S.	987654321	1941-06-20	291 Berry,Bellaire,TX	4	Administration	987654321
Narayan,Ramesh K.	666884444	1962-09-15	975 FireOak,Humble,TX	5	Research	333445555
English,Joyce A.	453453453	1972-07-31	5631 Rice,Houston,TX	5	Research	333445555
Jabbar,Ahmad V.	987987987	1969-03-29	980 Dallas,Houston,TX	4	Administration	987654321
Borg,James E.	888665555	1937-11-10	450 Stone,Houston,TX	1	Headquarters	888665555

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
123456789	1	32.5	Smith,John B.	ProductX	Bellaire
123456789	2	7.5	Smith,John B.	ProductY	Sugarland
666884444	3	40.0	Narayan,Ramesh K.	ProductZ	Houston
453453453	1	20.0	English,Joyce A.	ProductX	Bellaire
453453453	2	20.0	English,Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong,Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong,Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong,Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong,Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya,Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya,Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar,Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar,Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace,Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace,Jennifer S.	Reorganization	Houston
888665555	20	null	Borg,James E.	Reorganization	Houston

Note: The above figure is now called Figure 10.4 in Edition 4

Guideline to Redundant Information in Tuples and Update Anomalies

- **GUIDELINE 2:** Design a schema that does not suffer from the insertion, deletion and update anomalies. If there are any present, then note them so that applications can be made to take them into account

1.3 Null Values in Tuples

GUIDELINE 3: Relations should be designed such that their tuples will have as few NULL values as possible

- Attributes that are NULL frequently could be placed in separate relations (with the primary key)
- Reasons for nulls:
  - attribute not applicable or invalid
  - attribute value unknown (may exist)
  - value known to exist, but unavailable

1.4 Spurious Tuples

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- The "lossless join" property is used to guarantee meaningful results for join operations

GUIDELINE 4: The relations should be designed to satisfy the lossless join condition. No spurious tuples should be generated by doing a natural-join of any relations.

Q. Consider the relation r shown below and its decompositions R_1 and R_2. Assume that $X \rightarrow Y$ and $Z \rightarrow Y$. Is this decomposition lossless or lossy?

r

X	Y	Z
x_1	y_1	z_1
x_2	y_2	z_2
x_3	y_2	z_3
x_4	y_3	z_4

R_1

X	Y
x_1	y_1
x_2	y_2
x_3	y_2
x_4	y_3

R_2

Y	Z
y_1	z_1
y_2	z_2
y_2	z_3
y_3	z_4

R_1 Join R_2

X	Y	Z	
x_1	y_1	z_1	✓
x_2	y_2	z_2	✓
x_2	y_2	z_3	Spurious Tuples
x_3	y_2	z_2	Spurious Tuples
x_3	y_2	z_3	✓
x_4	y_3	z_4	✓

Spurious Tuples = They represent False Information.

Spurious Tuples (2)

There are two important properties of decompositions:

(a) non-additive or losslessness of the corresponding join

(b) preservation of the functional dependencies.

Note that property (a) is extremely important and *cannot* be sacrificed. Property (b) is less stringent and may be sacrificed. (See Chapter 11).

2.1 Functional Dependencies (1)

- Functional dependencies (FDs) are used to specify *formal measures* of the "goodness" of relational designs
- FDs and keys are used to define **normal forms** for relations
- FDs are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes
- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y

Functional Dependencies (2)

- X -> Y holds if whenever two tuples have the same value for X, they *must have* the same value for Y
- For any two tuples t1 and t2 in any relation instance r(R): *If* t1[X]=t2[X], *then* t1[Y]=t2[Y]
- X -> Y in R specifies a *constraint* on all relation instances r(R)
- Written as X -> Y; can be displayed graphically on a relation schema as in Figures. (denoted by the arrow:).
- FDs are derived from the real-world constraints on the attributes

Definition of FD

- A **functional dependency**, denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of R specifies a *constraint* on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$, they must also have $t_1[Y] = t_2[Y]$.

Examples of FD constraints (1)

- social security number determines employee name

  SSN -> ENAME
- project number determines project name and location

  PNUMBER -> {PNAME, PLOCATION}
- employee ssn and project number determines the hours per week that the employee works on the project

  {SSN, PNUMBER} -> HOURS

Examples of FD constraints (2)

- An FD is a property of the attributes in the schema R
- The constraint must hold on *every relation instance* r(R)
- If K is a key of R, then K functionally determines all attributes in R (since we never have two distinct tuples with t1[K]=t2[K])

2.2 Inference Rules for FDs (1)

- Given a set of FDs F, we can *infer* additional FDs that hold whenever the FDs in F hold

Armstrong's inference rules:

IR1. (**Reflexive**) If Y *subset-of* X, then X -> Y

IR2. (**Augmentation**) If X -> Y, then XZ -> YZ

(Notation: XZ stands for X ∪ Z)

IR3. (**Transitive**) If X -> Y and Y -> Z, then X -> Z

- IR1, IR2, IR3 form a *sound* and *complete* set of inference rules

Inference Rules for FDs (2)

Some **additional inference rules** that are useful:

(**Decomposition**) If X -> YZ, then X -> Y and X -> Z

(**Union**) If X -> Y and X -> Z, then X -> YZ

(**Psuedotransitivity**) If X -> Y and WY -> Z, then WX -> Z

- The last three inference rules, as well as any other inference rules, can be deduced from IR1, IR2, and IR3 (completeness property)

Example of FD

Figure 15.8
A relation *R*(A, B, C, D) with its extension.

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

Continued

- See the illustrative example relation in Figure 15.8. Here, the following FDs *may hold* because the four tuples in the current extension have no violation of these constraints:
- *B* □ *C; holds*
- *C* □ *B; holds*
- {*A, B*}□ *C; holds*
- {*A, B*}□ *D;* and
- {*C, D*} □ *B. holds*
- However, the following *do not* hold because we already have violations of them in the given extension*: A* □ *B*
- (tuples 1 and 2 violate this constraint); *B* □ *A* (tuples 2 and 3 violate this constraint);
- *D* □ *C* (tuples 3 and 4 violate it).

Continued

- A □ A always holds
- B □ B always holds
- C □ C always holds
- D □ D always holds
- The LHS of the dependency is known as Determinant.

Exercise Time…….Are u ready?

Solve the question in next slide.

15.26. Consider the following relation:

A	B	C	TUPLE#
10	b1	c1	1
10	b2	c2	2
11	b4	c1	3
12	b3	c4	4
13	b1	c1	5
14	b3	c4	6

a. Given the previous extension (state), which of the following dependencies *may hold* in the above relation? If the dependency cannot hold, explain why *by specifying the tuples that cause the violation.*

i. $A \rightarrow B$, ii. $B \rightarrow C$, iii. $C \rightarrow B$, iv. $B \rightarrow A$, v. $C \rightarrow A$

Inference Rules for FDs (3)

- **Closure** of a set F of FDs is the set F^+ of all FDs that can be inferred from F
- **Closure** of a set of attributes X with respect to F is the set X^+ of all attributes that are functionally determined by X
- X^+ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F

Q. Consider the relation r shown below and its decompositions R_1 and R_2. Assume that $X \rightarrow Y$ and $Z \rightarrow Y$. Is this decomposition lossless or lossy?

r

X	Y	Z
x_1	y_1	z_1
x_2	y_2	z_2
x_3	y_2	z_3
x_4	y_3	z_4

R_1

X	Y
x_1	y_1
x_2	y_2
x_3	y_2
x_4	y_3

R_2

Y	Z
y_1	z_1
y_2	z_2
y_2	z_3
y_3	z_4

R_1 Join R_2

X	Y	Z	
x_1	y_1	z_1	✓
x_2	y_2	z_2	✓
x_2	y_2	z_3	Spurious Tuples
x_3	y_2	z_2	Spurious Tuples
x_3	y_2	z_3	✓
x_4	y_3	z_4	✓

Spurious Tuples = They represent False Information.

Q. Finding closure of Given set of FD set F?

Given $F = \{A \rightarrow B, C \rightarrow D\}$ with $C \subseteq B$, show that $F \models A \rightarrow D$.

Solution as $C \subseteq B$ $\therefore B \rightarrow C$ holds. [Reflexive Rule] ..... (1)

Given $A \rightarrow B$

$B \rightarrow C$ [Transitivity Rule]

$\therefore A \rightarrow C$

Given $C \rightarrow D$

using $A \rightarrow C$ and $C \rightarrow D$

$\therefore \boxed{A \rightarrow D}$ Hence proved.

Q: Given the set $F = \{ A \rightarrow B, C \rightarrow X, BX \rightarrow Z \}$ derive $AC \rightarrow Z$ using inference axioms.

Solution :-

$$F = \{ A \rightarrow B \quad -(1)$$
$$C \rightarrow X \quad -(2)$$
$$BX \rightarrow Z \} \quad -(3)$$

using (1) and (3) pseudotransitivity Rule

$$A \rightarrow B$$
$$BX \rightarrow Z$$
$$\therefore AX \rightarrow Z \quad \cdots (4)$$

$$C \rightarrow X \quad -(2)$$
$$AX \rightarrow Z \quad -(4)$$

$$\therefore AC \rightarrow Z$$

Q: Given the relational scheme R(A, B, C, D) and FDs $A \rightarrow B$ and $BC \rightarrow D$. Determine which of the functional dependencies shown below can be derived from these FDs by application of the inference axioms.

(a) $Ac \rightarrow D$ (b) $B \rightarrow D$ (c) $AD \rightarrow B$

① $A \rightarrow B$ — (1)
$Bc \rightarrow D$ — (2) pseudo Transitivity Rule

$\boxed{Ac \rightarrow D}$

② $B \rightarrow D$ can not be determined.

③ $A \rightarrow B$

$AD \rightarrow B \cdot D$ [augmentation with D on Both side]

$\boxed{AD \rightarrow B}$

Armstrong's axioms

- Armstrong's axioms are **sound**, because they do not generate any incorrect functional dependencies.
- They are **complete**, because, for a given set F of functional dependencies,

  they allow us to generate all F^+.

Finding closure of an attribute set

Must Read

```
result := α;
while (changes to result) do
    for each functional dependency β → γ in F do
        begin
            if β ⊆ result then result := result ∪ γ;
        end
```

Figure 7.7 An algorithm to compute α^+, the closure of α under F.

Finding closure of an attribute set

- There are several uses of the attribute closure algorithm:
- To test if α is a superkey, we compute α^+, and check if α^+ contains all attributes of R.

problems:- $R=(A, B, C, G, H, I)$ and set of functional dependency

$F=\{A \rightarrow B$
$A \rightarrow C$
$CG \rightarrow H$
$CG \rightarrow I$
$B \rightarrow H\}$

Find $(AG)^+$

means finding closure of attribute set (AG).

result := AG
result := ABG
result := ABCG
result := ABCGH
result := ABCGHI

$\therefore (AG)^+ := ABCGHI$

$\because A \subseteq AG$

$\because A \subseteq AG$
$\because CG \subseteq ABCG$
$\because CG \subseteq ABCGH$

AG is superkey for relation R.

(2) $F = \{ SSn \rightarrow Ename$

$Pnumber \rightarrow \{Pname, Plocation\},$

$\{SSn, Pnumber\} \rightarrow Hours \}$

Compute $\{ssn\}^+$

$\{Pnumber\}^+$

$\{SSn, Pnumber\}^+$

(i) $\{SSn\}^+$

result := {ssn}

result := {ssn, Ename}

(ii) $\{Pnumber\}^+$

result := {Pnumber}

result := {Pnumber, Pname, Plocation}

(iii) $\{ssn, Pnumber\}^+$

result := {ssn, Pnumber}

result := {ssn, Pnumber, Ename}

result := {ssn, Pnumber, Ename, Pname, Plocation, Hours}

(3) Compute the B^+ ?

Given $R=(A,B,C,D,E)$

$F=\{A\to BC$
$CD\to E$
$B\to D$
$E\to A\}$

List the candidate keys of R ?

(i) B^+ ?

result := {B}
result := {BD}

(ii) $(CD)^+$??

result := {CD}
result := {CDE}
result := {ACDE}
result := {ABCDE}

(iii) $(ABE)^+$

result := {ABE}
result := {ABCE}
result := {ABCDE}
$\therefore (ABE)^+ = \{ABCDE\}$

(iv) $(AC)^+$

result := {AC}
result := {ABC}
result := {ABCD}
$(AC)^+ := \{ABCDE\}$

(v) $(AB)^+$

result := {AB}
result := {ABC}
result := {ABCD}
result := {ABCDE}
$(AB)^+ := \{ABCDE\}$

Candidate Key of Relation R are :

(AB), (CD), (AC)

2.3 Equivalence of Sets of FDs

- Two sets of FDs F and G are **equivalent** if:
  - every FD in F can be inferred from G, *and*
  - every FD in G can be inferred from F
- Hence, F and G are equivalent if $F^+ = G^+$

Definition: F **covers** G if every FD in G can be inferred from F (i.e., if G^+ *subset-of* F^+)

- F and G are equivalent if F covers G and G covers F
- There is an algorithm for checking equivalence of sets of FDs

Equivalence of Sets of FDs

- **Definition.**

Two sets of functional dependencies *E* and *F* are **equivalent** if $E^+ = F^+$. Therefore, equivalence means that every FD in *E* can be inferred from *F*, and every FD in *F* can be inferred from *E*; that is, *E* is equivalent to *F* if both the conditions—*E* covers *F* *and F* covers *E*—hold.

Equivalence of Sets of FDs example

- Given Set F={A □C
  AC □D
  E □AD
  E □H}

and set G={A □CD
E □AH}

Equivalence of functional dependency set:

Q: Given set $F = \{ A \to C,\ AC \to D,\ E \to AD,\ E \to H \}$ and $G = \{ A \to CD,\ E \to AH \}$

check whether set F and G are equivalent.

(1) $(A)^+$

result := {A}
:= {AC}
$(A)^+$:= {ACD}

$\therefore A \to CD$

(2) $(E)^+$

result := {E}
:= {AED}
:= {AEDH}
$(E)^+$:= {ACDEH}

$\therefore E \to AH$

(1) $(A)^+$ In set G.

result := {A}
result := {ACD}

$\therefore A \to C$

(2) $(E)^+$

result := {E}
result := {EAH}
$(E)^+$:= {EACDH}

$\therefore E \to AD$
$\therefore E \to H$

Prime and Non-Prime Attribute

An attribute of relation schema R is called a prime attribute of R, if it is a member of some candidate key of R.

An attribute is called non prime if it is not a prime attribute-i.e. if it is not a member of any candidate key.

Definitions of Keys and Attributes Participating in Keys (2)

- If a relation schema has more than one key, each is called a **candidate key.** One of the candidate keys is *arbitrarily* designated to be the **primary key,** and the others are called *secondary keys*.
- A **Prime attribute** must be a member of *some candidate key*
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

problem :- $R = (A, B, C, D)$

$F = \{ AB \rightarrow C, \; C \rightarrow D, \; D \rightarrow A \}$

are satisfied by a relation R.

1) What are the candidate keys of R.
2) What is the primary key of Relation R.
3) What are the prime attributes of Relation R.
4) Are there any super key for relation R.
5) What are the non-prime attributes in relation R.

(i) $(AB)^+ = ?$

result := {AB}
:= {ABC}
$(AB)^+ := \{ A B C D \}$

(ii) $(C)^+ = ?$

result := {C}
:= {CD}
:= {CDA}
$(C)^+ := \{ A C D \}$

(iii) $(D)^+ = ?$

result := {D}
result := {AD}
result := {AD}

(iv) $(AC)^+ = ?$

result := {Ac}
:= {ACD}
$(Ac)^+$:= {ACD}

(v) $(Bc)^+ := ?$

result := {Bc}
:= {BcD}
$(Bc)^+$:= {ABCD} ✓

(vi) $(BD)^+ = ?$

result := {BD}
:= {ABD}
$(BD)^+$:= {ABcD} ✓

1 Ans:- AB, BC, and BD.

2 Ans:- AB

3 Ans:- A, B, C, and D.

4 Ans:- {ABCD}, {ABC}, {ABD}

5 Ans:- No, there are no prime attributes in Relation R.

Full functional dependency

- **Full functional dependency** - a FD Y □ Z where removal of any attribute from Y means the FD does not hold any more

Examples: - {SSN, PNUMBER} -> HOURS is a full FD since neither

SSN -> HOURS

nor

PNUMBER -> HOURS hold

Partial functional dependency

- - {SSN, PNUMBER} □ ENAME is *not* a full FD (it is called a *partial dependency*) since
- SSN □ ENAME also holds

Non trivial Functional dependency

- If a functional dependency X->Y holds true where Y is not a subset of X then this dependency is called non trivial Functional dependency.
- **For example**:
  An employee table with three attributes: (emp_id, emp_name, emp_address).

- The following functional dependencies are non-trivial:
- emp_id -> emp_name
- (emp_name is not a subset of emp_id)

- emp_id -> emp_address
- (emp_address is not a subset of emp_id)

Trivial Functional dependency

- If a functional dependency X->Y holds true where Y is a subset of X then this dependency is called trivial Functional dependency.

- On the other hand, the following dependencies are trivial:
  {emp_id, emp_name} -> emp_name
- [emp_name is a subset of {emp_id, emp_name}]

Pitfalls in Relational-Database Design

- Before we continue our discussion of normal forms, let us look at what can go wrong in a bad database design. Among the undesirable properties that a bad design may have are:
- Repetition of information.
- Inability to represent certain information.

EXAMPLE OF AN UPDATE ANOMALY (1)

Consider the relation:

EMP_PROJ (Emp#, Proj#, Ename, Pname, No_hours)

- **Update Anomaly:** Changing the name of project number P1 from "Billing" to "Customer-Accounting" may cause this update to be made for all 100 employees working on project P1.

EXAMPLE OF AN UPDATE ANOMALY (2)

- **Insert Anomaly:** Cannot insert a project unless an employee is assigned to .

  Inversely - Cannot insert an employee unless an he/she is assigned to a project.

- **Delete Anomaly:** When a project is deleted, it will result in deleting all the employees who work on that project. Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

3 Normal Forms Based on Primary Keys

3.1 Normalization of Relations

3.2 Practical Use of Normal Forms

3.3 Definitions of Keys and Attributes Participating in Keys

3.4 First Normal Form

3.5 Second Normal Form

3.6 Third Normal Form

3.1 Normalization of Relations (1)

- **Normalization**: The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations.

- **Normal form**: Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form.

Normalization of Relations (2)

- 2NF, 3NF, BCNF based on keys and FDs of a relation schema
- 4NF based on keys, multi-valued dependencies : MVDs; 5NF based on keys, join dependencies : JDs (Chapter 11)
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation; Chapter 11)

3.2 Practical Use of Normal Forms

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are **hard to understand** or to **detect**
- The database designers ***need not*** normalize to the highest possible normal form. (usually up to 3NF, BCNF or 4NF)
- **Denormalization:** the process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

3.3 Definitions of Keys and Attributes Participating in Keys (1)

- A **superkey** of a relation schema $R = \{A_1, A_2, ..., A_n\}$ is a set of attributes S *subset-of* R with the property that no two tuples t_1 and t_2 in any legal relation state r of R will have $t_1[S] = t_2[S]$

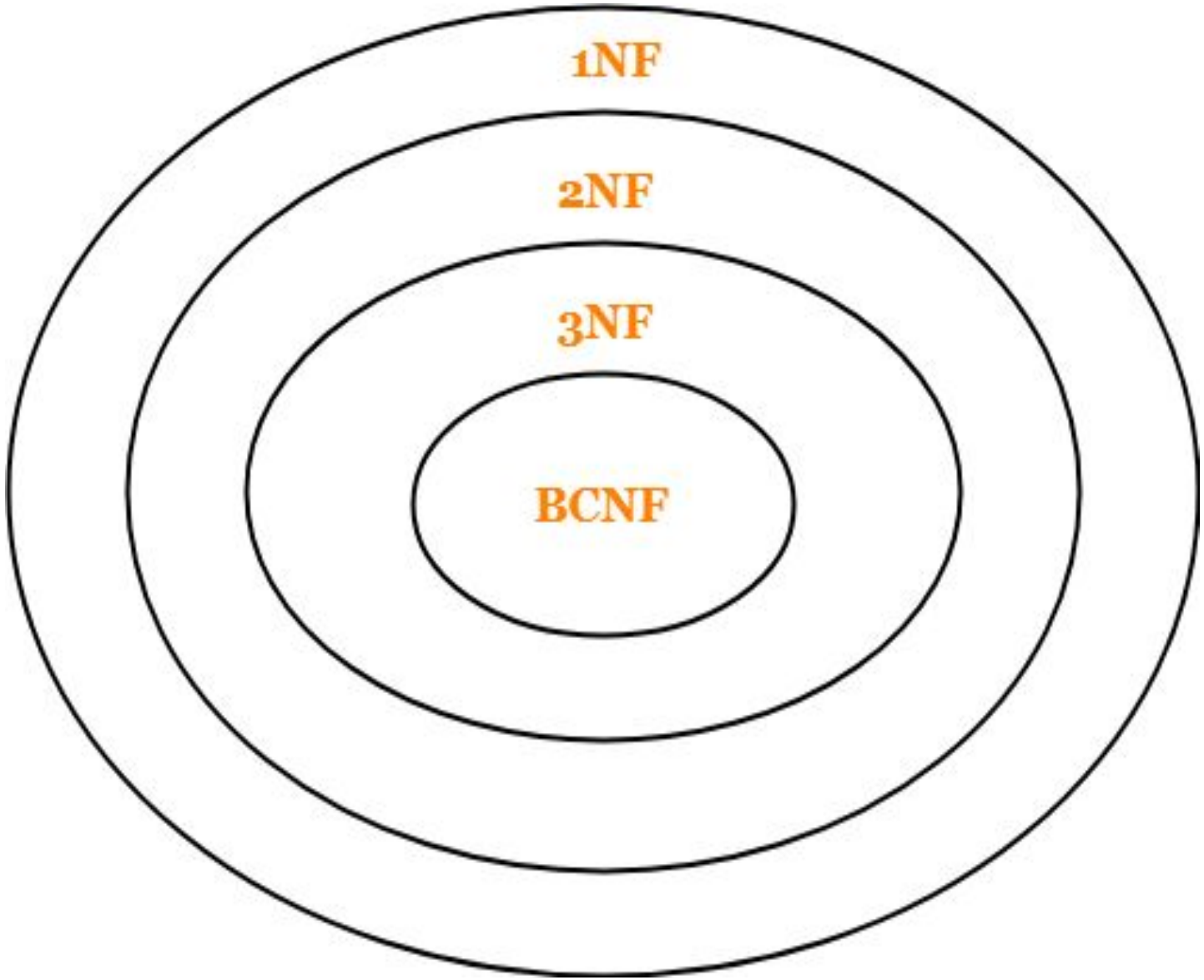
- A **key** K is a superkey with the *additional property* that removal of any attribute from K will cause K not to be a superkey any more.

Definitions of Keys and Attributes Participating in Keys (2)

- If a relation schema has more than one key, each is called a **candidate key.** One of the candidate keys is *arbitrarily* designated to be the **primary key,** and the others are called *secondary keys*.
- A **Prime attribute** must be a member of *some candidate key*
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

Normal Forms

- We now present the first three normal forms: 1NF, 2NF, and 3NF. These were proposed by Codd (1972) as a sequence to achieve the desirable state of 3NF relations.
- progressing through the intermediate states of 1NF,2NF ,and 3NF if needed.



3.2 First Normal Form

- A relation schema R is said to be in first normal form if and only if ,the domains of all the attributes of R are atomic(i.e. indivisible)
- It states that the domain of an attribute must include only atomic (simple,indivisible)values and that the value of any attribute in a tuple must be a single value from the domain of the attribute.

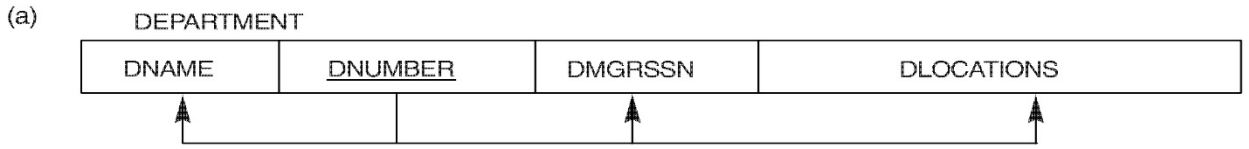
3.2 First Normal Form

- Disallows composite attributes, multivalued attributes, and **nested relations**; attributes whose values *for an individual tuple* are non-atomic

- Considered to be part of the definition of relation

Figure 10.8 Normalization into 1NF

Figure 14.8 Normalization into 1NF. (a) Relation schema that is not in 1NF. (b) Example relation instance. (c) 1NF relation with redundancy.



(b)

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Note: The above figure is now called Figure 10.8 in Edition 4

Figure 14.9 Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a "nested relation" PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposing EMP_PROJ into 1NF relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

(a) **EMP_PROJ**

SSN	ENAME	PROJS	
		PNUMBER	HOURS

(b) **EMP_PROJ**

SSN	ENAME	PNUMBER	HOURS
123456789	Smith,John B.	1	32.5
		2	7.5
666884444	Narayan,Ramesh K.	3	40.0
453453453	English,Joyce A.	1	20.0
		2	20.0
333445555	Wong,Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya,Alicia J.	30	30.0
		10	10.0
987987987	Jabbar,Ahmad V.	10	35.0
		30	5.0
987654321	Wallace,Jennifer S.	30	20.0
		20	15.0
888665555	Borg,James E.	20	null

(c) **EMP_PROJ1**

SSN	ENAME

EMP_PROJ2

SSN	PNUMBER	HOURS

3.3 Second Normal Form (1)

- Uses the concepts of **FD**s, **primary key**

Definitions:

- **Prime attribute** - attribute that is member of the primary key K
- **Full functional dependency** - a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more

Examples: - {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold

- {SSN, PNUMBER} -> ENAME is *not* a full FD (it is called a *partial dependency*) since SSN -> ENAME also holds

Second Normal Form (2)

- A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on the primary key
- No partial functional dependency should Exist in relation R.
- R can be decomposed into 2NF relations via the process of 2NF normalization

General Normal Form Definitions (For Multiple Keys) (1)

- The above definitions consider the primary key only.
- The following more general definitions take into account relations with multiple candidate keys
- A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on *every key* of R

Normal Forms

Q.1 Consider the relation $r(A, B, C, D, E)$ and the set of functional dependencies is

$F = \{ AB \rightarrow CE,\ E \rightarrow AB,\ C \rightarrow D \}$

Q. What is the highest normal form of this relation r?

Q. What are the prime and non-prime attributes?

Hint :- To determine the normal form of the Given relation, we need to determine all possible keys of the relation.

$(AB)^+ = ?$
result := {AB}
:= {ABCE}
$(AB)^+ := \{ABCDE\}$ ✓

$(C)^+ := ?$
result := {C}
$(C)^+ := \{CD\}$

$(E)^+ := ?$
result := {E}
:= {EAB}
:= {EABC}
$(E)^+ := \{ABCDE\}$ ✓

Solution :- Since r is a relation, by the definition of relation, r is in 1NF.

- All non-prime attributes are fully functionally dependent on primary key of R.
- $AB \rightarrow C$, $C \rightarrow D$ [Transitive Dependency]

- The Highest Normal Form is 2NF.
- A, B, and E are prime attributes.
- C and D are non-prime —||—

Q.2 What normal form are the following relations in and why?

Sample-I (H, I, J, K, L, M, N, O)

Only {H, I} can act as the key.

functional dependencies are :-

$F = \{ H, I \rightarrow J, K, L$
$J \rightarrow M$
$K \rightarrow N$
$L \rightarrow O \}$

[As per Transitivity Rule]

$H, I \rightarrow J$
$J \rightarrow M$
$\therefore H, I \rightarrow M$

$H, I \rightarrow K$
$K \rightarrow N$
$\therefore H, I \rightarrow N$

$H, I \rightarrow L$
$L \rightarrow O$
$\therefore H, I \rightarrow O$

$HI \subseteq HI$ [Reflexive rule]
$\therefore HI \rightarrow HI$

$\therefore HI \rightarrow H, I, J, K, L, M, N, O$

The relation Sample-I is in 2NF, bcoz all non prime attributes are fully functionally dependent on the key {H, I}.

(ii) Sample-2 (A, B, C, D, E, F, G)

Functional Dependencies are

$F = \{ A, B \rightarrow C, D, E, F, G$

$C, F \rightarrow A$

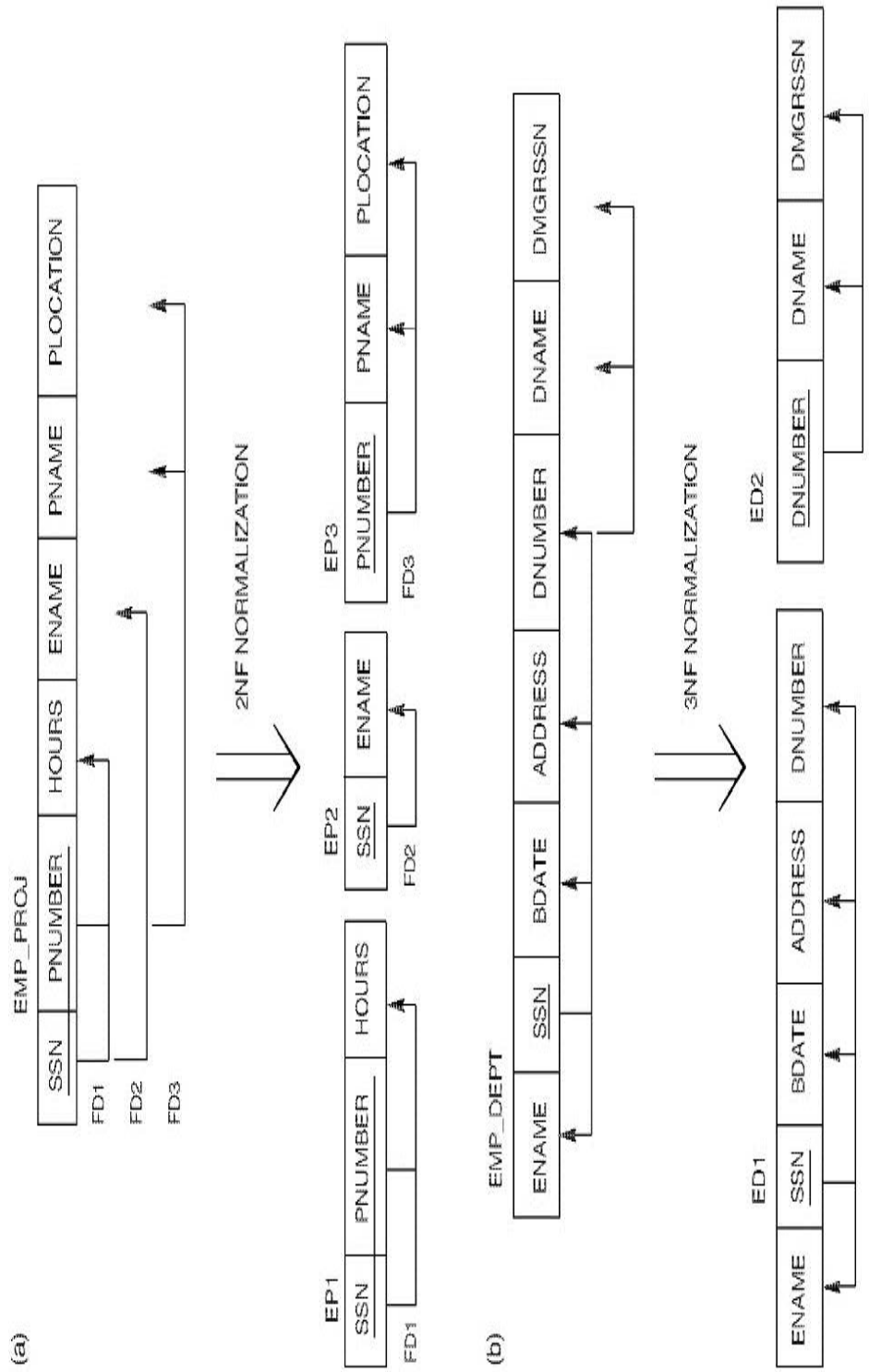
$A \rightarrow C \}$

$A, B \rightarrow C$ } partial dependency is present;

$A \rightarrow C$ } B is an extraneous attribute

- The Highest Normal form of this relation is 1NF.
- It is not in 2NF, bcoz partial dependency exist in relation sample-2.

Figure 14.10 The normalization process. (a) Normalizing EMP_PROJ in 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.



3.4 Third Normal Form (1)

Definition:

- **Transitive functional dependency** - a FD X -> Z that can be derived from two FDs X -> Y and Y -> Z

  Examples:

  - SSN -> DMGRSSN is a *transitive* FD since SSN -> DNUMBER and DNUMBER -> DMGRSSN hold

  - SSN -> ENAME is *non-transitive* since there is no set of attributes X where SSN -> X and X -> ENAME

Third Normal Form (2)

- A relation schema R is in **third normal form (3NF)** if it is in 2NF *and* no non-prime attribute A in R is transitively dependent on the primary key
- R can be decomposed into 3NF relations via the process of 3NF normalization

NOTE:

In X -> Y and Y -> Z, with X as the primary key, we consider this a problem only if Y is not a candidate key. When Y is a candidate key, there is no problem with the transitive dependency .

E.g., Consider EMP (SSN, Emp#, Salary).

Here, SSN -> Emp# -> Salary and Emp# is a candidate key.

General Normal Form Definitions (2)

Definition (very imp)

- **Superkey** of relation schema R - a set of attributes S of R that contains a key of R
- A relation schema R is in **third normal form** (**3NF**) if whenever a FD X **->** A holds in R, then either:

  (a) X is a superkey of R, or

  (b) A is a prime attribute of R

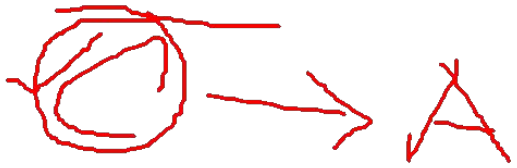
NOTE: Boyce-Codd normal form disallows condition (b) above

Guidelines for 3NF

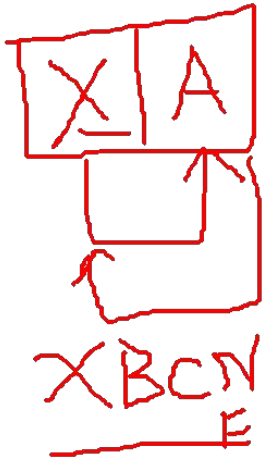
- No Transitive functional dependency should in relation R.(imp rule)
- One Non-Prime attribute should not determine another non prime attribute. (imp rule)

BCNF (Boyce- Codd Normal Form)

- A relation schema R is in **Boyce-Codd Normal Form** (**BCNF**) if whenever an FD X -> A holds in R, then X is a superkey of R.



- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF



- There exist relations that are in 3NF but not in BCNF
- The goal is to have each relation in BCNF (or 3NF)

Q. Consider the relation scheme R(A, B, C) with FDs

F = {AB → C
C → A}

Show that the scheme R is in 3NF, but not in BCNF.

also, determine the minimal key of R.

R(A, B, C)

AB → C

C → A

i) $(AB)^+ = ?$

result := {AB}

:= {ABC}

∴ AB is key of R.

ii) $(C)^+ = ?$

result := {C}

result := {AC}

The relation R is in 3NF, bcoz there are no transitive dependencies and in FD C → A, A is a prime attribute.

The Relation R is not in BCNF, bcoz C → A, this FD violates the properties / rules of BCNF.

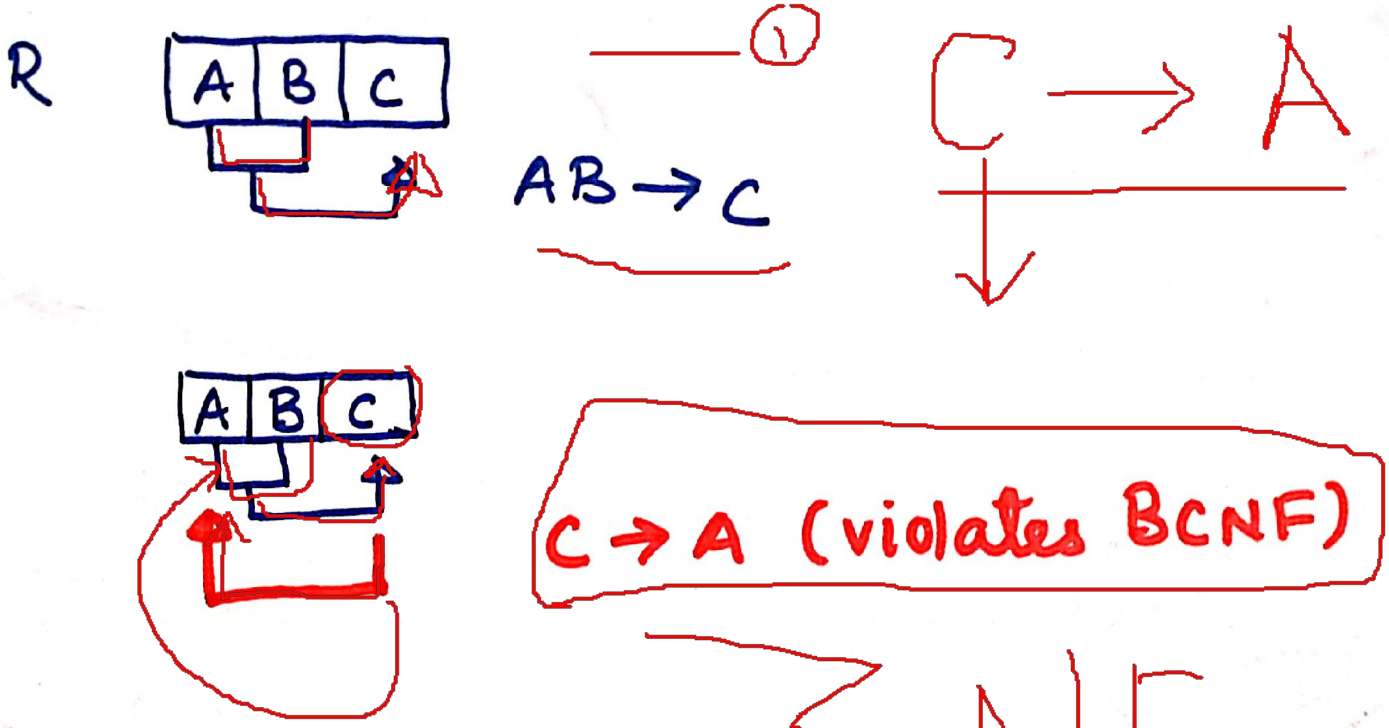


Figure 14.11 Normalization to 2NF and 3NF. (a) The lots relation schema and its functional dependencies fd1 through FD4. (b) Decomposing lots into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of normalization of lots.

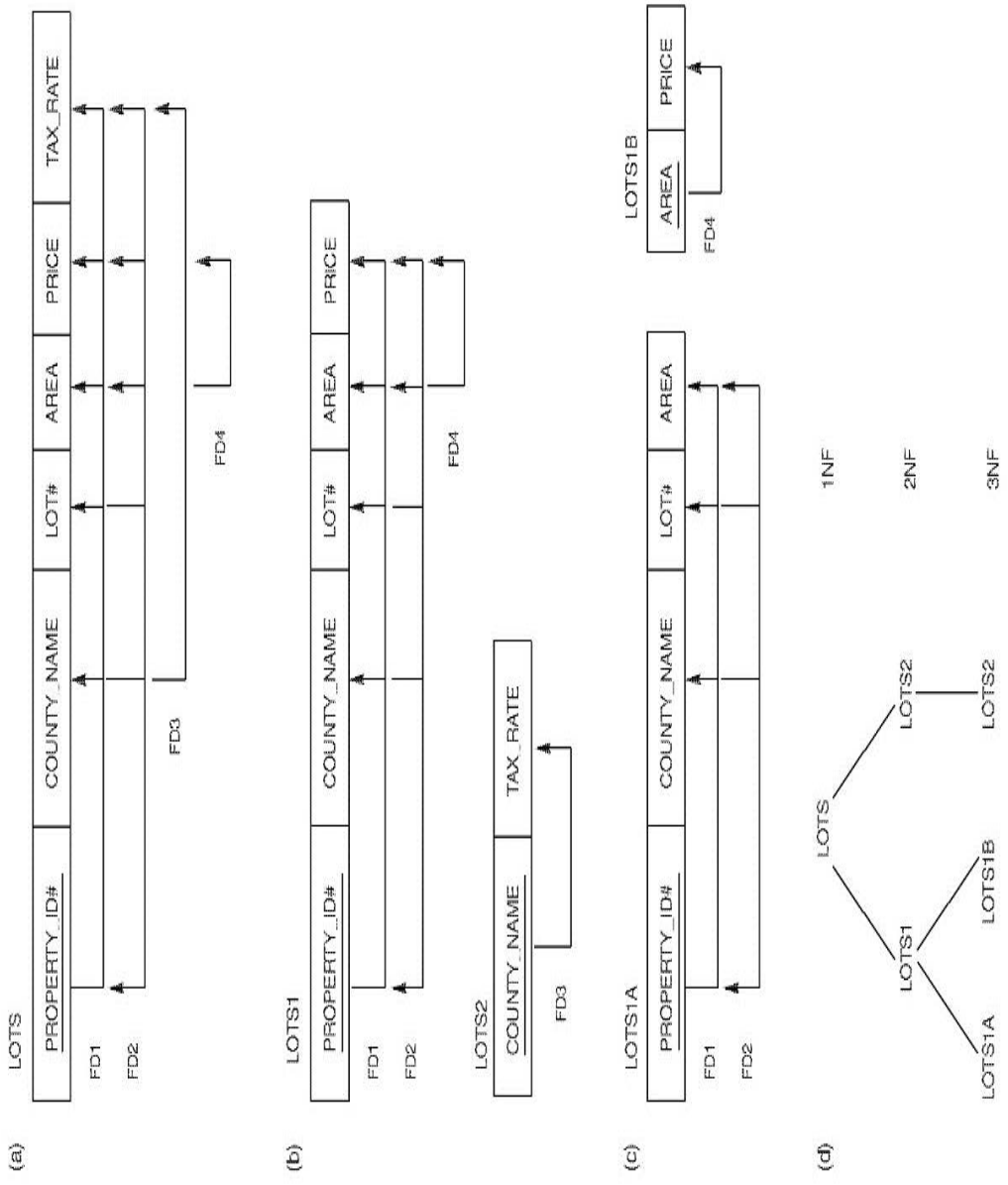
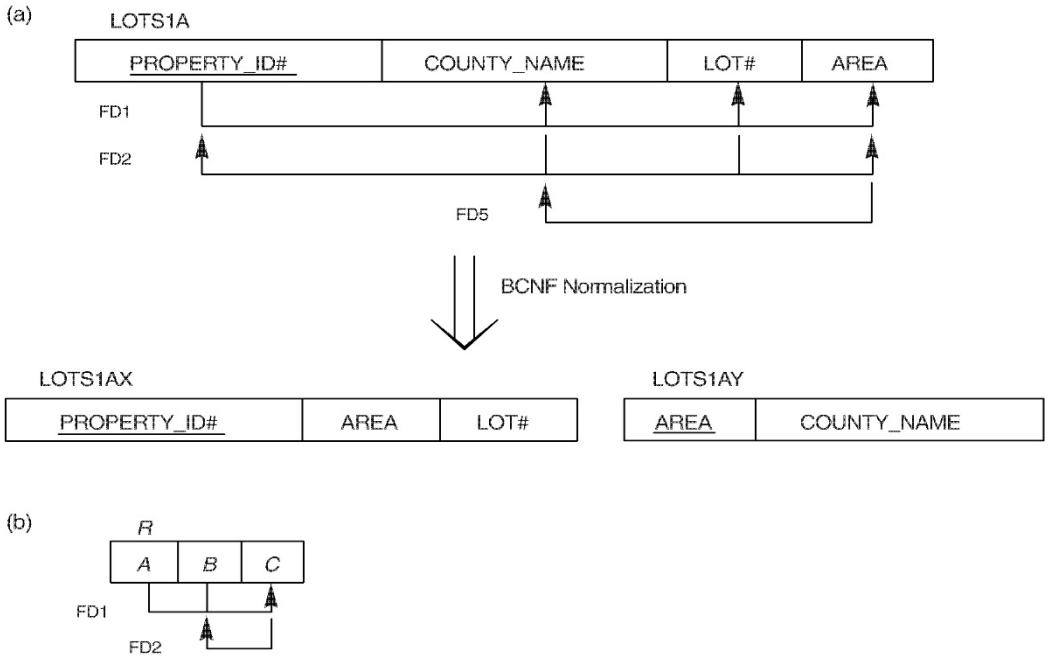


Figure 10.12 Boyce-Codd normal form

Figure 14.12 Boyce-Codd normal form. (a) BCNF normalization with the dependency of FD2 being "lost" in the decomposition. (b) A relation *R* in 3NF but not in BCNF.



Note: The above figure is now called Figure 10.12 in Edition 4

Figure 10.13 a relation TEACH that is in 3NF but not in BCNF

Figure 14.13 A relation TEACH that is in 3NF but not in BCNF.

TEACH

STUDENT	COURSE	INSTRUCTOR
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe

Note: The above figure is now called Figure 10.13 in Edition 4

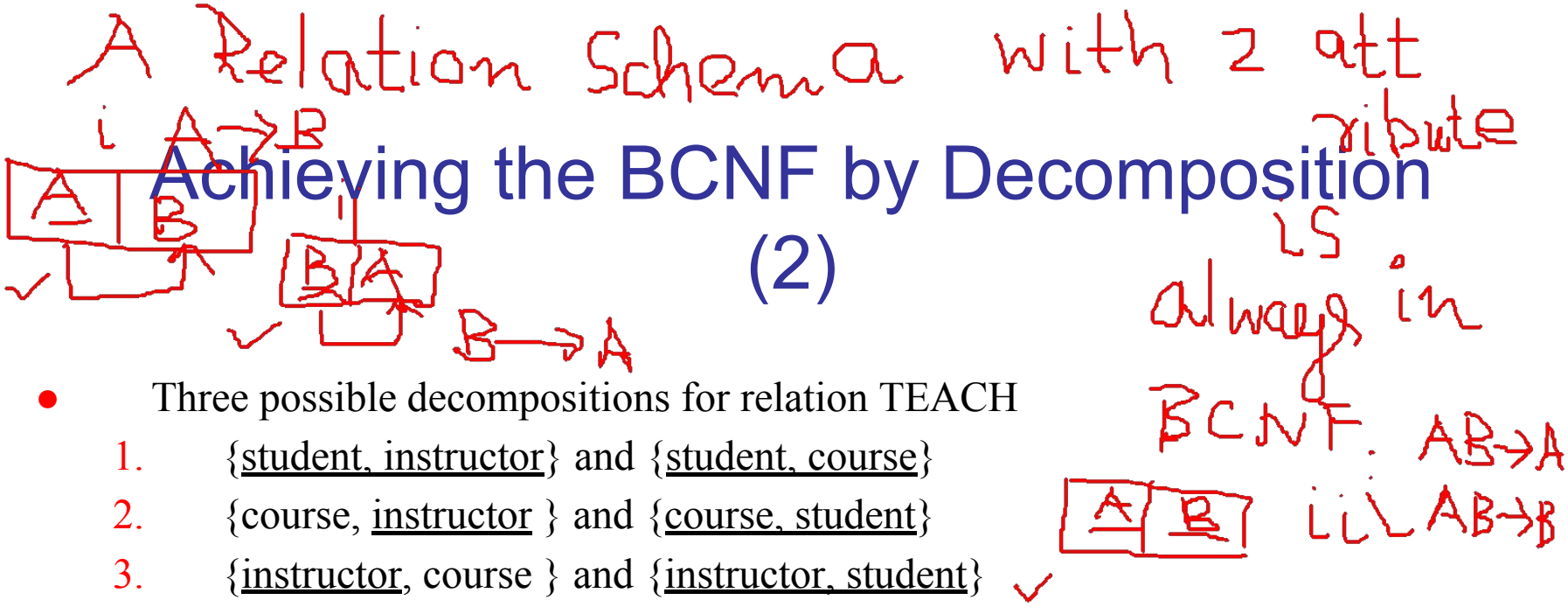
Achieving the BCNF by Decomposition (1)

- Two FDs exist in the relation TEACH:

  fd1: { student, course} -> instructor

  fd2: instructor -> course
- {student, course} is a candidate key for this relation and that the dependencies shown follow the pattern in Figure 10.12 (b). So this relation is in 3NF but not in BCNF
- A relation **NOT** in BCNF should be decomposed so as to meet this property, while possibly forgoing the preservation of all functional dependencies in the decomposed relations. (See Algorithm 11.3)

Achieving the BCNF by Decomposition (2)



- Three possible decompositions for relation TEACH
  1. {student, instructor} and {student, course}
  2. {course, instructor } and {course, student}
  3. {instructor, course } and {instructor, student}
- All three decompositions will lose fd1. We have to settle for sacrificing the functional dependency preservation. But we cannot sacrifice the non-additivity property after decomposition.
- Out of the above three, only the 3rd decomposition will not generate spurious tuples after join.(and hence has the non-additivity property).
- A test to determine whether a binary decomposition (decomposition into two relations) is nonadditive (lossless) is discussed in section 11.1.4 under Property LJ1. Verify that the third decomposition above meets the property.

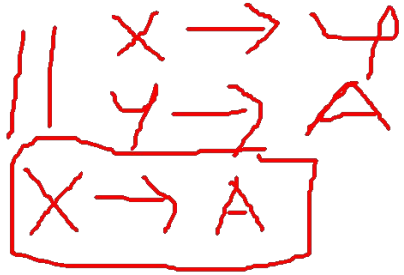
2.4 Minimal Sets of FDs (1)

- A set of FDs is **minimal** if it satisfies the following conditions:

(1) Every dependency in F has a single attribute for its RHS.

(2) We cannot remove any dependency from F and have a set of dependencies that is equivalent to F.

(3) We cannot replace any dependency X -> A in F with a dependency Y -> A, where Y proper-subset-of X (Y subset-of X) and still have a set of dependencies that is equivalent to F.



Minimal Sets of FDs (2)

- Every set of FDs has an equivalent minimal set
- There can be several equivalent minimal sets
- There is no simple algorithm for computing a minimal set of FDs that is equivalent to a set F of FDs
- To synthesize a set of relations, we assume that we start with a set of dependencies that is a minimal set (e.g., see algorithms 11.2 and 11.4)

Minimal Sets of FDs

A set of functional dependencies F to be minimal if it satisfies the following conditions:

1. Every dependency in F has a single attribute for its right-hand side.

2. We cannot replace any dependency $X \rightarrow A$ in F with a dependency $Y \rightarrow A$,where Y is a proper subset of X, and still have a set of dependencies that is equivalent to F.

3. We cannot remove any dependency from F and still have a set of dependencies that is equivalent to F.

We can think of a minimal set of dependencies as being a set of dependencies in a standard or canonical form and with no redundancies.

Algorithm 16.2. Finding a Minimal Cover F for a Set of Functional Dependencies E

Input: A set of functional dependencies E.

1. Set $F := E$.
2. Replace each functional dependency $X \rightarrow \{A_1, A_2, ..., A_n\}$ in F by the n functional dependencies $X \rightarrow A_1, X \rightarrow A_2, ..., X \rightarrow A_n$.
3. For each functional dependency $X \rightarrow A$ in F
   for each attribute B that is an element of X
   if $\{ \{F - \{X \rightarrow A\}\} \cup \{ (X - \{B\}) \rightarrow A\} \}$ is equivalent to F
   then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ in F.
4. For each remaining functional dependency $X \rightarrow A$ in F
   if $\{F - \{X \rightarrow A\}\}$ is equivalent to F,
   then remove $X \rightarrow A$ from F.

We illustrate the above algorithm with the following:

Let the given set of FDs be $E: \{B \rightarrow A, D \rightarrow A, AB \rightarrow D\}$. We have to find the minimal cover of E.

- All above dependencies are in canonical form (that is, they have only one attribute on the right-hand side), so we have completed step 1 of Algorithm 16.2 and can proceed to step 2. In step 2 we need to determine if $AB \rightarrow D$ has any redundant attribute on the left-hand side; that is, can it be replaced by $B \rightarrow D$ or $A \rightarrow D$?

- Since B $\rightarrow$ A, by augmenting with B on both sides (IR2), we have $BB \rightarrow AB$, or $B \rightarrow AB$ (i). However, $AB \rightarrow D$ as given (ii).
- Hence by the transitive rule (IR3), we get from (i) and (ii), $B \rightarrow D$. Thus $AB \rightarrow D$ may be replaced by $B \rightarrow D$.
- We now have a set equivalent to original E, say E': $\{B \rightarrow A, D \rightarrow A, B \rightarrow D\}$. No further reduction is possible in step 2 since all FDs have a single attribute on the left-hand side.
- In step 3 we look for a redundant FD in E'. By using the transitive rule on $B \rightarrow D$ and $D \rightarrow A$, we derive $B \rightarrow A$. Hence $B \rightarrow A$ is redundant in E' and can be eliminated.
- Therefore, the minimal cover of E is $\{B \rightarrow D, D \rightarrow A\}$.

Canonical Cover

- Suppose that we have a set of functional dependencies *F* on a relation schema. Whenever a user performs an update on the relation, the database system must ensure that the update does not violate any functional dependencies, that is, all the functional dependencies in *F* are satisfied in the new database state.
- The system must roll back the update if it violates any functional dependencies in the set *F*.
- We can reduce the effort spent in checking for violations by testing a simplified set of functional dependencies that has the same closure as the given set. Any database that satisfies the simplified set of functional dependencies will also satisfy the original set, and vice versa, since the two sets have the same closure. However, the simplified set is easier to test.

Canonical Cover

The canonical cover of F, F_c, can be shown to have the same closure as F; hence, testing whether F_c is satisfied is equivalent to testing whether F is satisfied. However, F_c is minimal in a certain sense—it does not contain extraneous attributes, and it

$F_c = F$
repeat
  Use the union rule to replace any dependencies in F_c of the form
    $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ with $\alpha_1 \rightarrow \beta_1 \beta_2$.
  Find a functional dependency $\alpha \rightarrow \beta$ in F_c with an extraneous
    attribute either in α or in β.
    /* Note: the test for extraneous attributes is done using F_c, not F */
  If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$.
until F_c does not change.

Figure 7.8 Computing canonical cover

Consider the following set F of functional dependencies on schema (A, B, C):

$$A \rightarrow BC$$
$$B \rightarrow C$$
$$A \rightarrow B$$
$$AB \rightarrow C$$

Let us compute the canonical cover for F.

- There are two functional dependencies with the same set of attributes on the left side of the arrow:

  $$A \rightarrow BC$$
  $$A \rightarrow B$$

  We combine these functional dependencies into $A \rightarrow BC$.

- A is extraneous in $AB \rightarrow C$ because F logically implies $(F - \{AB \rightarrow C\}) \cup \{B \rightarrow C\}$. This assertion is true because $B \rightarrow C$ is already in our set of functional dependencies.

- C is extraneous in $A \rightarrow BC$, since $A \rightarrow BC$ is logically implied by $A \rightarrow B$ and $B \rightarrow C$.

Thus, our canonical cover is

$$A \rightarrow B$$
$$B \rightarrow C$$