Chapter 11

Functional Dependencies

Introduction

- Functional dependency (FD) is a relationship from one set of attributes to another
- For examples:
  - In Supplier-Part-Project (SPJ), the set of attributes {S#, P#, J#} → the set of attribute {QTY}
  - For any given values for the pair of attributes {S#, P#, J#}, there is just one corresponding value of attribute {QTY}, but
  - Many distinct values of {S#, P#, J#} can have the same corresponding value for attribute QTY

Basic Definitions

SCP

S#	CITY	P#	QTY
S1	London	P1	100
S1	London	P2	100
S2	Paris	P1	200
S2	Paris	P2	200
S3	Paris	P2	300
S4	London	P2	400
S4	London	P4	400
S4	London	P5	400

Definition of FD

- Let r be a relation, and let X and Y be arbitrary subsets of the set of attributes of r. Then we say that Y is functionally dependent on X

  $X \rightarrow Y$

  ("X functionally determines Y") if and only if each X value in r has associated with it precisely on one Y value in r. In other words, whenever two tuples of r agree on their X value, they also agree on their Y value.

  $\{S\#\} \rightarrow \{CITY\}$

  Because every tuple of that relation with a given S# value also has the same CITY value.

Example of FDs

SCP TABLE:

{S#, P#} → QTY

{S#, P#} → CITY

{S#, P#} → {CITY, QTY}

{S#, P#} → S#

{S#, P#} → {S#, P#, CITY, QTY}

{S#} → {CITY}

- If X is a candidate key of relvar R, the all attributes Y of relvar R must be functionally dependent on X

  {P#} → {P#, PNAME, COLOR, WEIGHT, CITY}

Trivial and Nontrivial Dependencies

- A FD is trivial if an only if the right-hand side is a subset of the left-hand side

  {S#, P#} → {S#}
- Nontrivial
- We are only interested in nontrivial FDs

Closure of A Set of Dependencies

- Some FDs might imply others

  {S#, P#} → {CITY, QTY}

  Implies

  {S#, P#} → {CITY}

  {S#, P#} → {QTY}

- Closure:
  - The set of all FDs that implied by a given set S of FDs is called the closure of S
  - Armstrong's Axioms
    - Then new FDs can be inferred from given ones

Inference Rules

1. Reflexivity: If B is a subset of A, then A → B
2. Augmentation: If A → B, then AC → BC
3. Transitivity: If A → B and B → C, then A → C
4. Self-determination: A → A
5. Decomposition: If A → BC, then A → B and A → C
   (Prove)
6. Union: If A → B and A → C, then A → BC
   (Prove)
7. Composition: If A → B and C → D, then AC → BD

Example:

{A} → {B,C}
{B} → {E}
{C,D} → {E, F}

Closure of S:

1. {A} → {B, C} (given)
2. {A} → {C} (decomposition)
3. {A, D} → {C, D} (augmentation)
4. {C, D} → {E, F} (given)
5. {A, D} → {E, F} (3 and 4, transitivity)
6. {A, D} → {F} (5, decomposition)

Irreducible Sets of Dependencies

A set S of FDs is irreducible if an only if it satifies the following three properties:

- The right-hand side of every FD in S involves just one attribute (singleton set)
- The left-hand side (determinant) of every FD in S is irreducible – meaning that no attribute can be discarded from the determinant without changing the closure S(+). It is called left-irreducible
- No FD in S can be discarded from S without changing the closure S (+).

Example

Irreducible

{P#} → {PNAME}

{P#} → {COLOR}

{P#} → {WEIGHT}

{P#} → {CITY}

Not irreducible

{P#, PNAME} → {COLOR}

{P#} → {PNAME}

{P#} → {WEIGHT}

{P#} → {CITY}

Not irreducible

{P#} → {PNAME, COLOR}

{P#} → {WEIGHT}

{P#} → {CITY}

Not irreducible

{P#} → {P#}

{P#} → {PNAME}

{P#} → {COLOR}

{P#} → {WEIGHT}

{P#} → {CITY}

Example

$A \rightarrow BC$

$B \rightarrow C$

$A \rightarrow B$

$AB \rightarrow C$

$AC \rightarrow D$

$A \rightarrow B$

$A \rightarrow C$

$B \rightarrow C$

$A \rightarrow B$

$AB \rightarrow C$

$AC \rightarrow D$

Irreducible set

$A \rightarrow B$

$B \rightarrow C$

$A \rightarrow D$

Chapter 12

Further Normalization I:
1NF, 2NF, 3NF, BCNF

Introduction

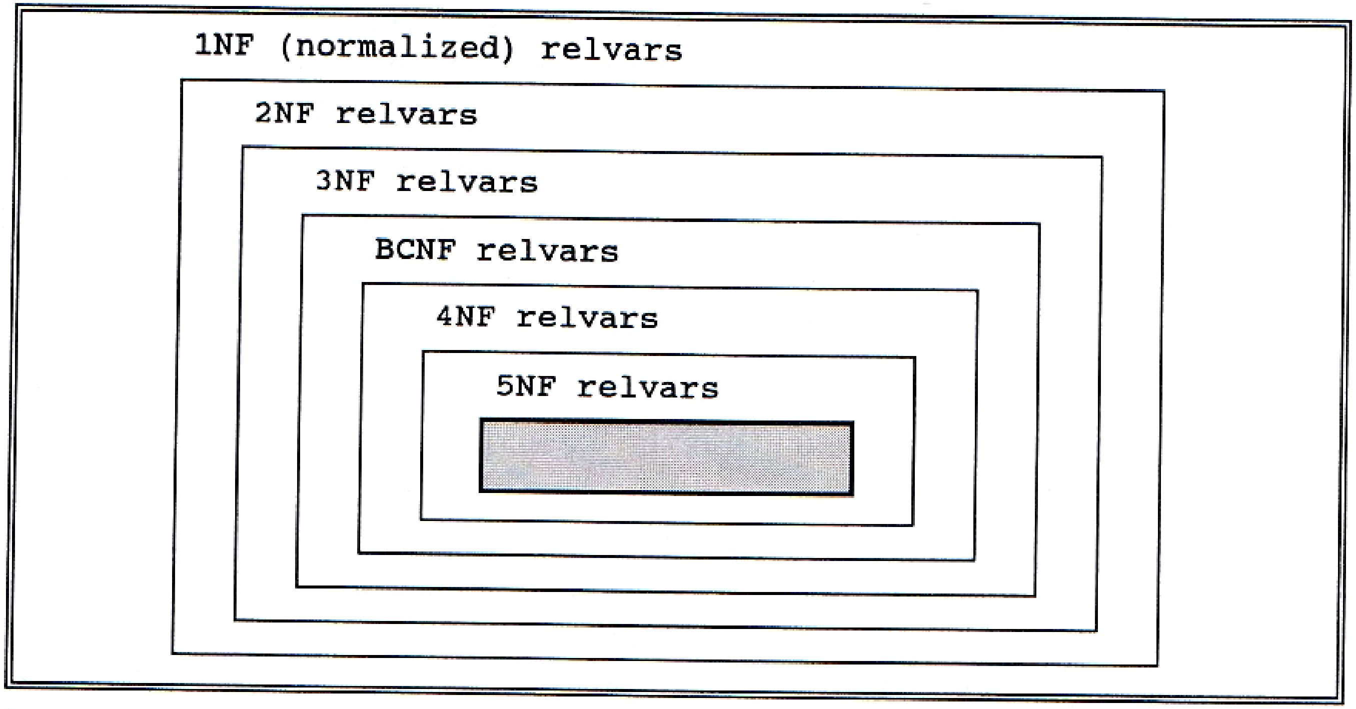
SCP

S#	CITY	P#	QTY
S1	London	P1	300
S1	London	P2	200
S1	London	P3	400
S1	London	P4	200
S1	London	P5	100
S1	London	P6	100
S2	Paris	P1	300
S2	Paris	P2	400
S3	Paris	P2	200
S4	London	P2	200
S4	London	P4	300
S4	London	P5	400

Introduction

- Redundancy
- “One fact in one place” (normalization)
- First normal form (1NF)
  - Relations are always in 1NF
  - Recall the property of relation: each tuple contains exactly one value for each attribute
- Normal forms
  - A relvar is said to be in a particular normal form if it satisfies a certain prescribed set of conditions
  - 1NF
  - 2NF
  - 3NF
  - Boyce/Codd (BCNF) (more desirable)

Introduction



Nonloss/Lossy Decomposition

- Decomposition process is a process of projection
- Nonloss (case a)
  - If we join SST and SC back together again, we get back to the original S
  - Reversible
- Lossy (case b)
  - Cannot back to the original S
- How do we know if we get back to the original S
  - Heath's theorem: Let R{A, B, C} be a relvar, where A, B, and C are sets of attributes. If R satisfies the FD A $\rightarrow$ B, then R is equal to the join of its projection on {A, B} and {A, C}

Example

S

S#	STATUS	CITY
S3	30	Paris
S5	30	Athens

(a) SST

S#	STATUS
S3	30
S5	30

SC

S#	CITY
S3	Paris
S5	Athens

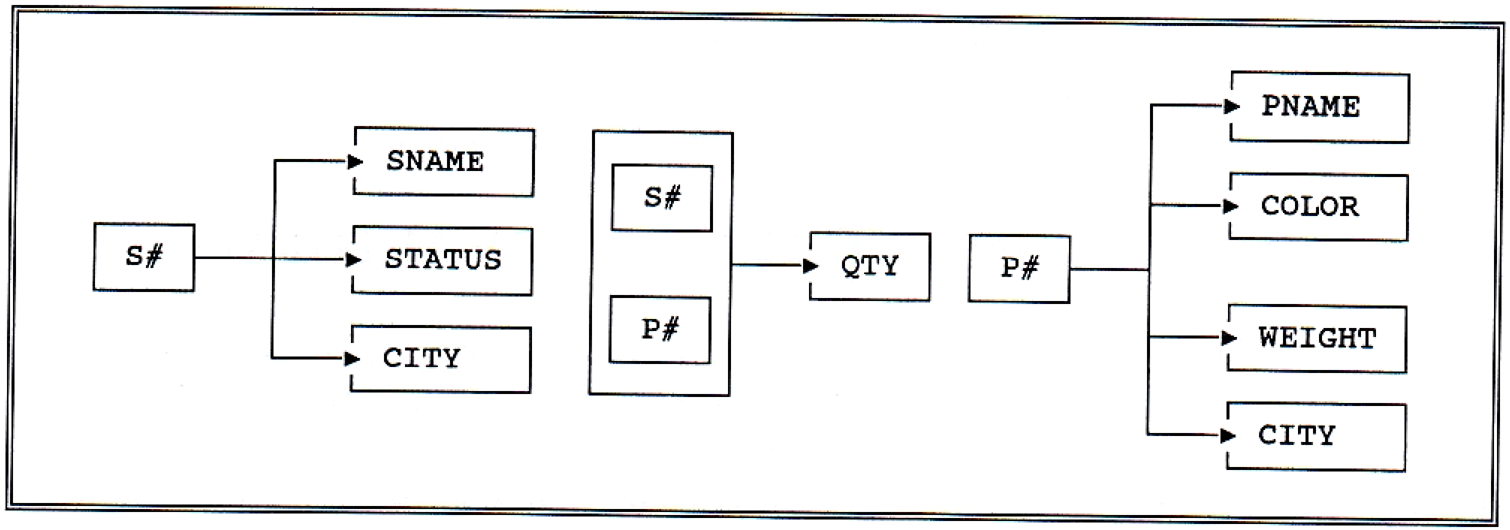
(b) SST

S#	STATUS
S3	30
S5	30

STC

STATUS	CITY
30	Paris
30	Athens

FD Diagram



The normalization is a procedure for eliminating arrows that are not arrows out of primary key

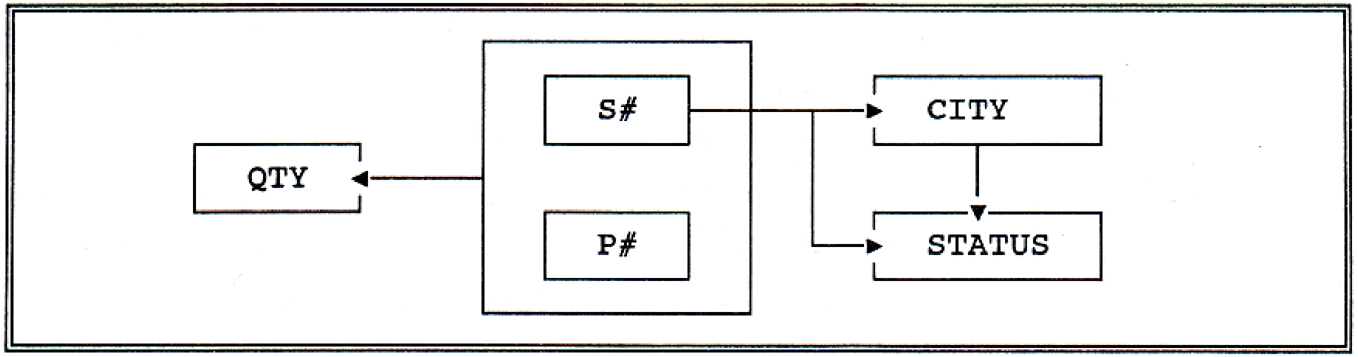
1NF, 2NF, 3NF Normal Forms

- First Norm Form
  - A relvar is in 1NF if and only if, in every legal value of that relvar, every tuple contains exactly value for each attribute

Why Normalization

FIRST {S#, STATUS, CITY, P#, QTY}
PRIMARY KEY {S#, P#}

CITY → STATUS



Sample Values of FIRST

FIRST

S#	STATUS	CITY	P#	QTY
S1	20	London	P1	300
S1	20	London	P2	200
S1	20	London	P3	400
S1	20	London	P4	200
S1	20	London	P5	100
S1	20	London	P6	100
S2	10	Paris	P1	300
S2	10	Paris	P2	400
S3	10	Paris	P2	200
S4	20	London	P2	200
S4	20	London	P4	300
S4	20	London	P5	400

Problems with FIRST

- INSERT
  - We cannot insert the fact that a particular supplier is located in a particular city until that supplier supplies at least one part
  - For example, S5 is located in Athens
  - Primary key {S#, P#} cannot be null
- DELETE
  - If we delete a tuple in FIRST for a supplier, we delete not only the shipment but also the information about the location (CITY). For example, {S3, P2}
  - FISRT contains too much information. We delete too much.
- UPDATE
  - For example, S1 moves from London to Amsterdam

Solutions to Problem

- Unbundling
  - SECOND {S#, STATUS, CITY} and SP {S#, P#, QTY}

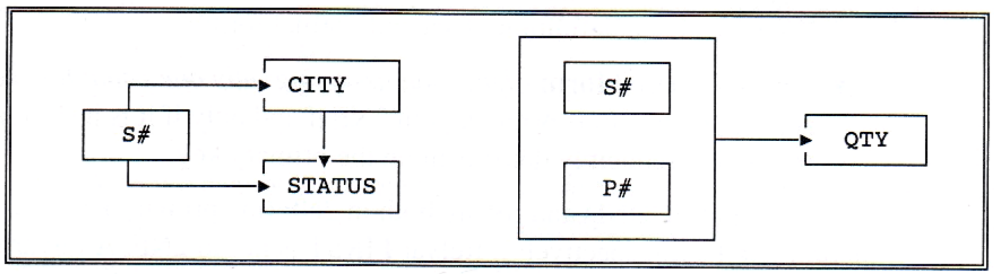


Fig. 11.7 FDs for relvars SECOND and SP

SECOND

S#	STATUS	CITY
S1	20	London
S2	10	Paris
S3	10	Paris
S4	20	London
S5	30	Athens

SP

S#	P#	QTY
S1	P1	300
S1	P2	200
S1	P3	400
S1	P4	200
S1	P5	100
S1	P6	100
S2	P1	300
S2	P2	400
S3	P2	200
S4	P2	200
S4	P4	300
S4	P5	400

2nd Normal Form

- Irreducible FDs:
  - For example: {S#, P#} $\rightarrow$ {CITY} in table SCP
  - P# is redundant for functional dependency purpose
  - Change into {S#} $\rightarrow$ {CITY}
  - CITY is irreducibly dependent on S#, but not irreducibly dependent on {S#, P#}
- A relvar is in 2NF if and only if it is in 1NF and every nonkey attribute is irreducibly dependent on the primary key (Assume only one candidate key)
  - Nonkey attribute is the attribute that does not participate in the primary key
  - SECOND and P are both in 2NF, FIRST is not in 2NF.

Problems with SECOND

- Lack of mutual independence among its nonkey attributes
- {S#} → {STATUS} is transitive via CITY

  {S#} → {CITY} → {STATUS}
- Transitive dependencies lead to "update" anomalies
  - INSERT: cannot insert the fact that a particular city has a particular status
  - DELETE: if we delete a tuple for a particular city, we delete not only supplier but also status
  - UPDATE: If we update status for London, we have to search every tuple with London

Solutions to Problem

- Break table SECOND into tables:

SC	{S#, CITY}
CS	{CITY, STATUS}

{S#} → {CITY}

{CITY} → {STATUS}

3rd Normal Form

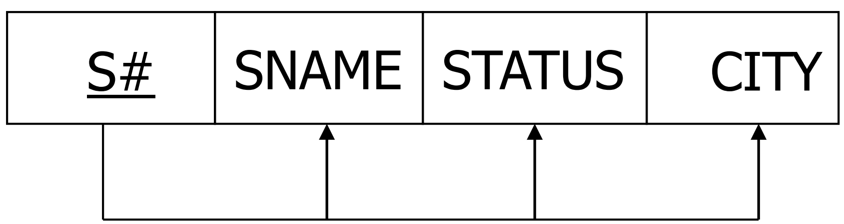
- A relvar is in 3NF if and only if it is in 2NF and every nonkey attribute is non-transitively dependent on the primary key.
  - Assume only on candidate key
  - Non-transitive dependency implies no mutual dependencies
  - SC and CS are both in 3NF but SECOND is not in 3NF

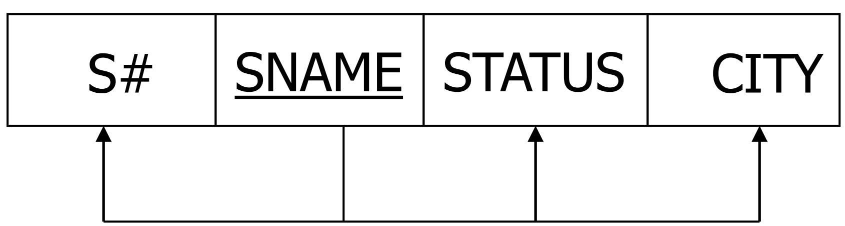
BOYCE/CODD Normal Form

- The previous NFs assume only one candidate key in a certain relvar
- Boyce/Codd Normal Form(BCNF): A relvar is in BCNF if and only if the determinants are candidate keys.
  - FIRST and SECOND are not in 3NF, BCNF
    - Among {S#}, {CITY}, and {S#, P#} in FIRST, only {S#, P#} is a candidate key
    - {CITY} in SECOND is not a candidate key
  - SP, SC, and CS are in 3NF, BCNF

Good Example

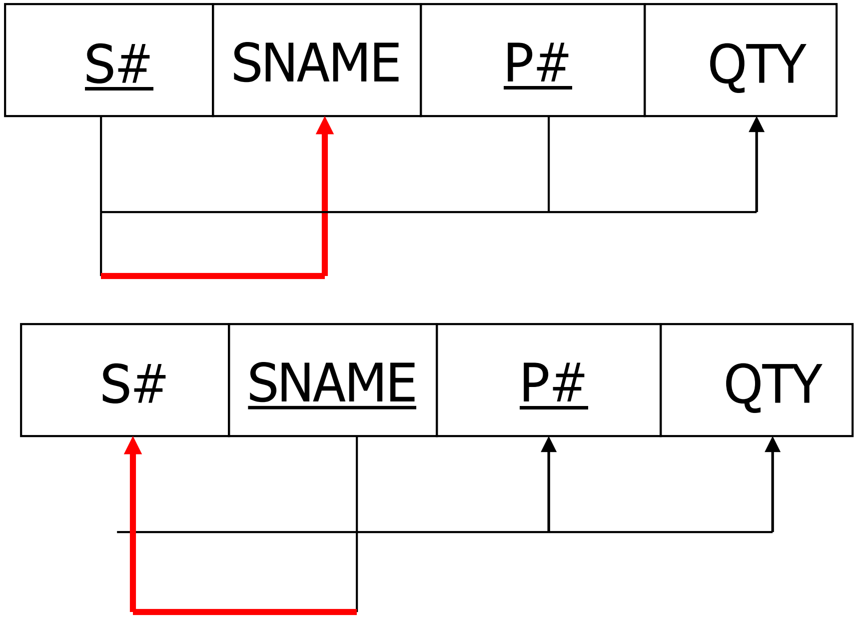
1. Assume in table S {S#, SNAME, STATUS, CITY}
    1. {S#} and {SNAME} are candidate keys
    2. {STATUS} and {CITY} are mutually independent



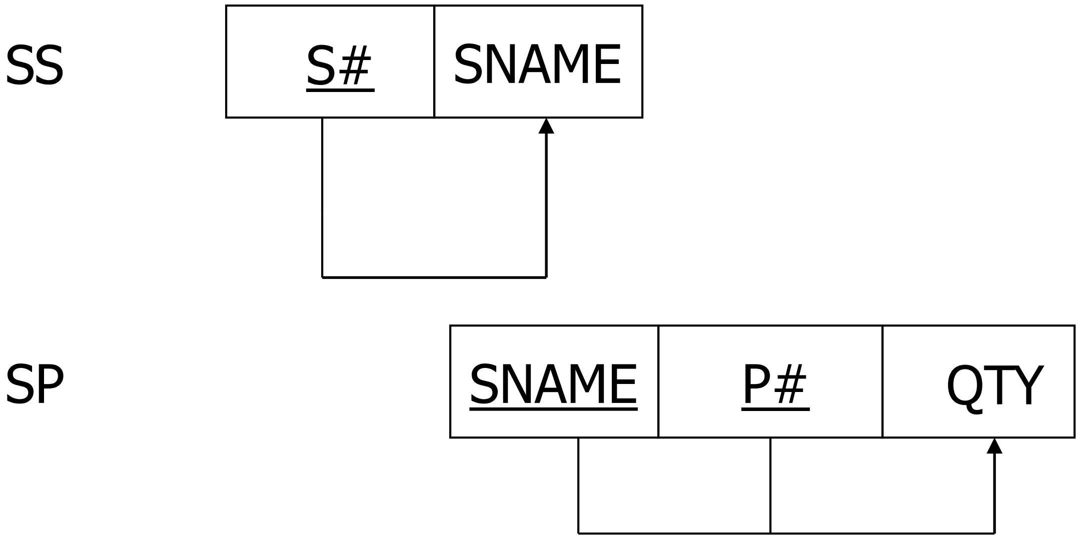


BCNF example

- SSP {S#, SNAME, P#, QTY}
  - Supplier name is unique
  - {S#, P#} and {SNAME, P#} are candidate keys
  - Is it in BCNF?



Solution



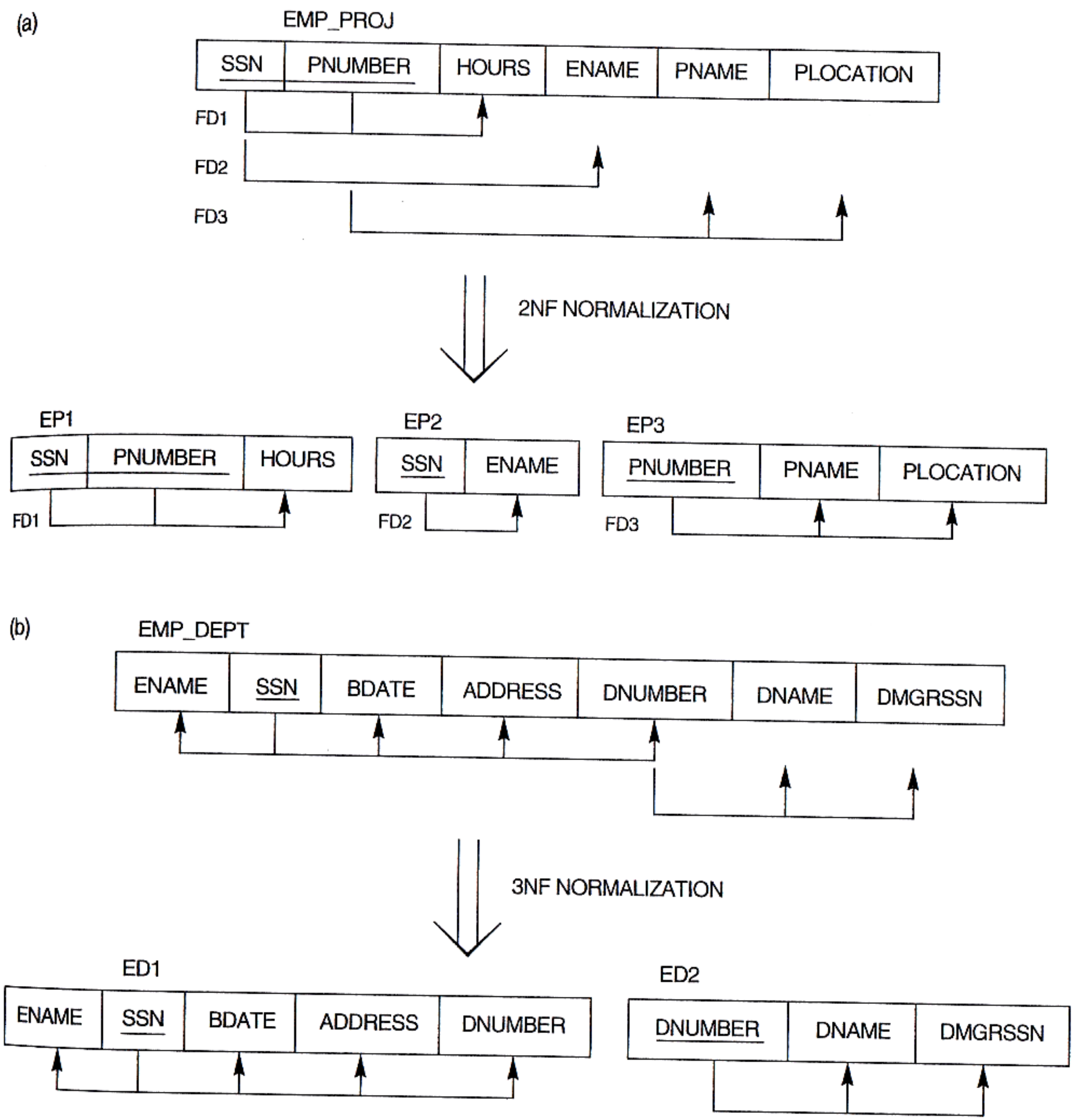
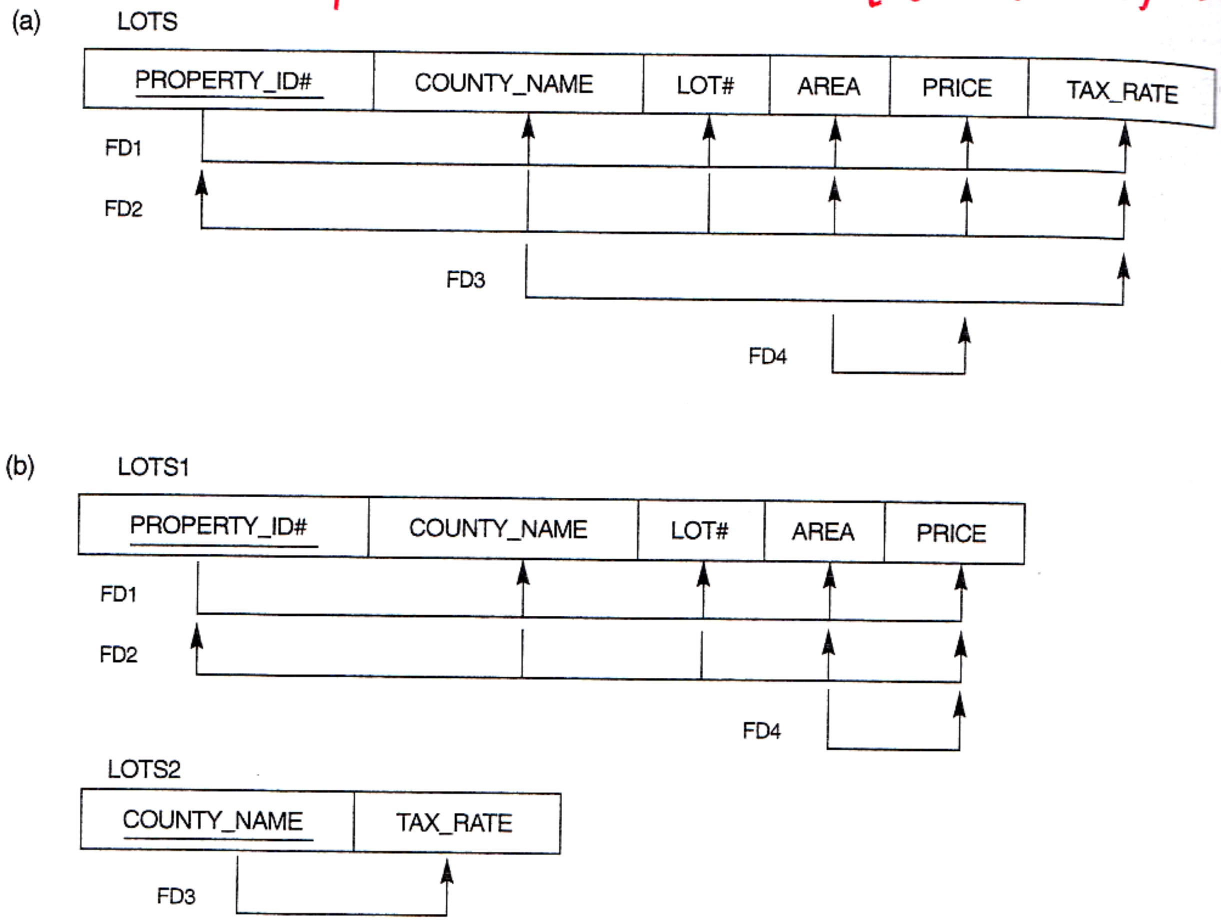


Figure 14.10 The normalization process. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

Candidate Keys : {PROPERTY_ID#} and {COUNTY_NAME, LOT#}



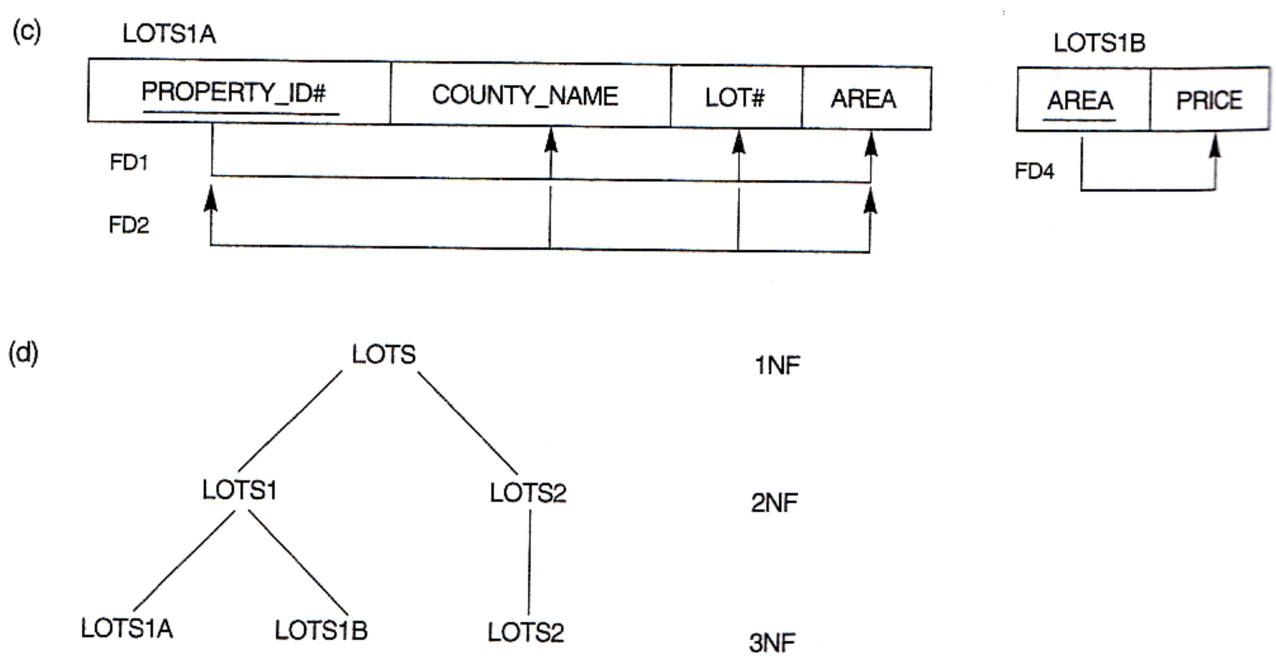


Figure 14.11 Normalization to 2NF and 3NF. (a) The LOTS relation schema and its functional dependencies FD1 through FD4. (b) Decomposing LOTS into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of normalization of LOTS.