Functional Dependencies

- Functional dependencies
  - Are used to specify *formal measures of the* "goodness" of relational designs
  - And keys are used to define **normal forms for** relations
  - 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

- 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.
- FDs are derived from the real world constraints on the attributes

Examples of FD constraints

- 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

- 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])

FD's are a property of the meaning of data and hold at all times: certain FD's can be ruled out based on a given state of the database

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

Figure
A relation state of TEACH with a *possible* functional dependency TEXT → COURSE. However, TEACHER → COURSE is ruled out.

Inference Rules for FDs

- 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 U Z)
  - IR3. (**Transitive) If X >**Y and Y >Z, then X >Z
- IR1, IR2, IR3 form a **sound and complete set of** inference rules
  - These are rules hold and all other rules that hold can be deduced from these

- 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

- **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

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 (**Covers):**
  - 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

Normalization of Relations

- **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

List of Normal Forms

- First Normal Form (1NF)
  - Atomic values
- 2NF, 3NF
  - based on primary keys
- 4NF
  - based on keys, multi-valued dependencies
- 5NF
  - based on keys, join dependencies

Practical Use of Normal Forms

- Most practical relational design projects take one of the following two approaches:
  - Perform a conceptual schema design using a conceptual model (ER, EER) and map the conceptual design into relations
  - Design the relations based on external knowledge derived from an existing implementation of files (or reports)

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties

- The database designers *need not* normalize to the highest possible normal form
  - (usually up to 3NF, BCNF or 4NF)

Definitions of Keys and Attributes Participating in Keys

- A **superkey** of a relation schema R = {A1, A2, ..., An} is a set of attributes S *subset-of* R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[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

- 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**

- 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.

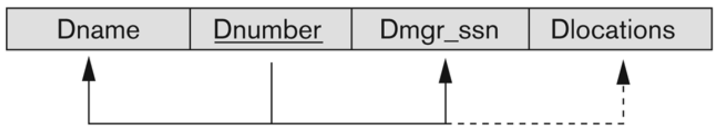
First Normal Form

- Historically, it is designed to disallow
  - composite attributes
  - multivalued attributes
  - Or the combination of both

- All the values need to be **atomic**

(a)

DEPARTMENT



(b)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

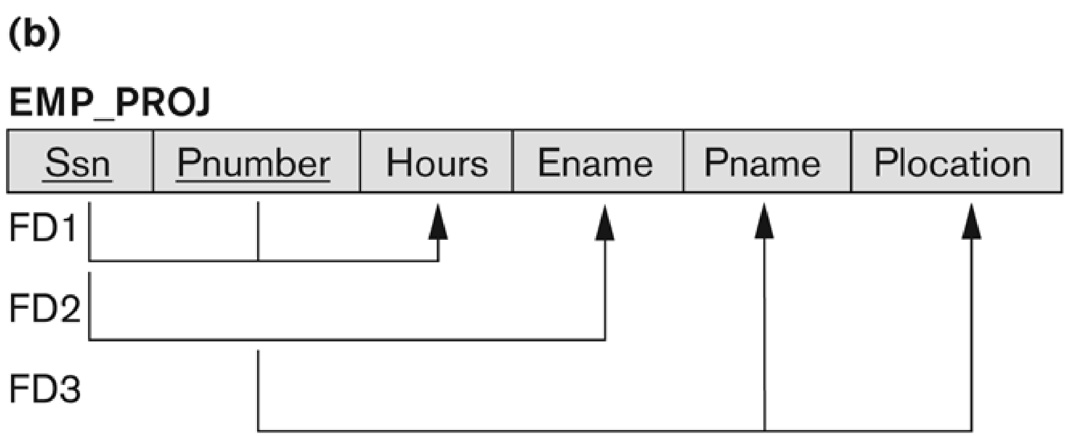
Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Figure
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

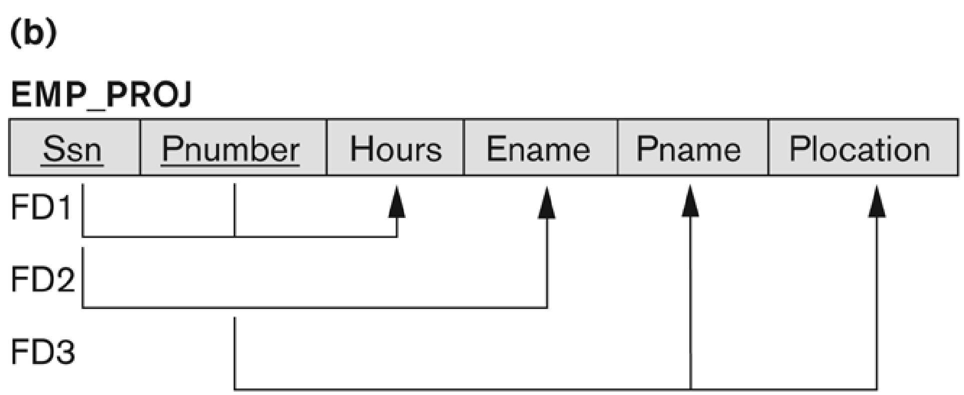
- To normalize into 1NF, we have the following 3 techniques:
  - Remove the attribute Dlocations that violates 1NF and place it in a separate relation
  - Expand the key (10.8 C). In this case, the PK become the combination of {Dnumber, Dlocation}
  - If the max number of values is known, then we can replace the violate attribute by the max number atomic attributes, such as, Dlocation1, Dlocation2, Dlocation3…

Second Normal Form

- In this example, {Ssn, Pnummber} -> Hours is a fully dependency
- However, the dependency {Ssn, Pnumber}->Ename is partial because Ssn->Ename holds



- 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
- A functional dependency X->Y is a **partial dependency** if some attribute A belong X can be removed from X and the dependency still holds



- If the primary key contains a **single attribute**, it is 2NF

- Normalization into 2NF:
  - If a relation schema is not in 2NF, it can be normalized into a number of 2NF relations where nonprime attributes are associated with only with the part of the primary key on which they are fully functionally dependent

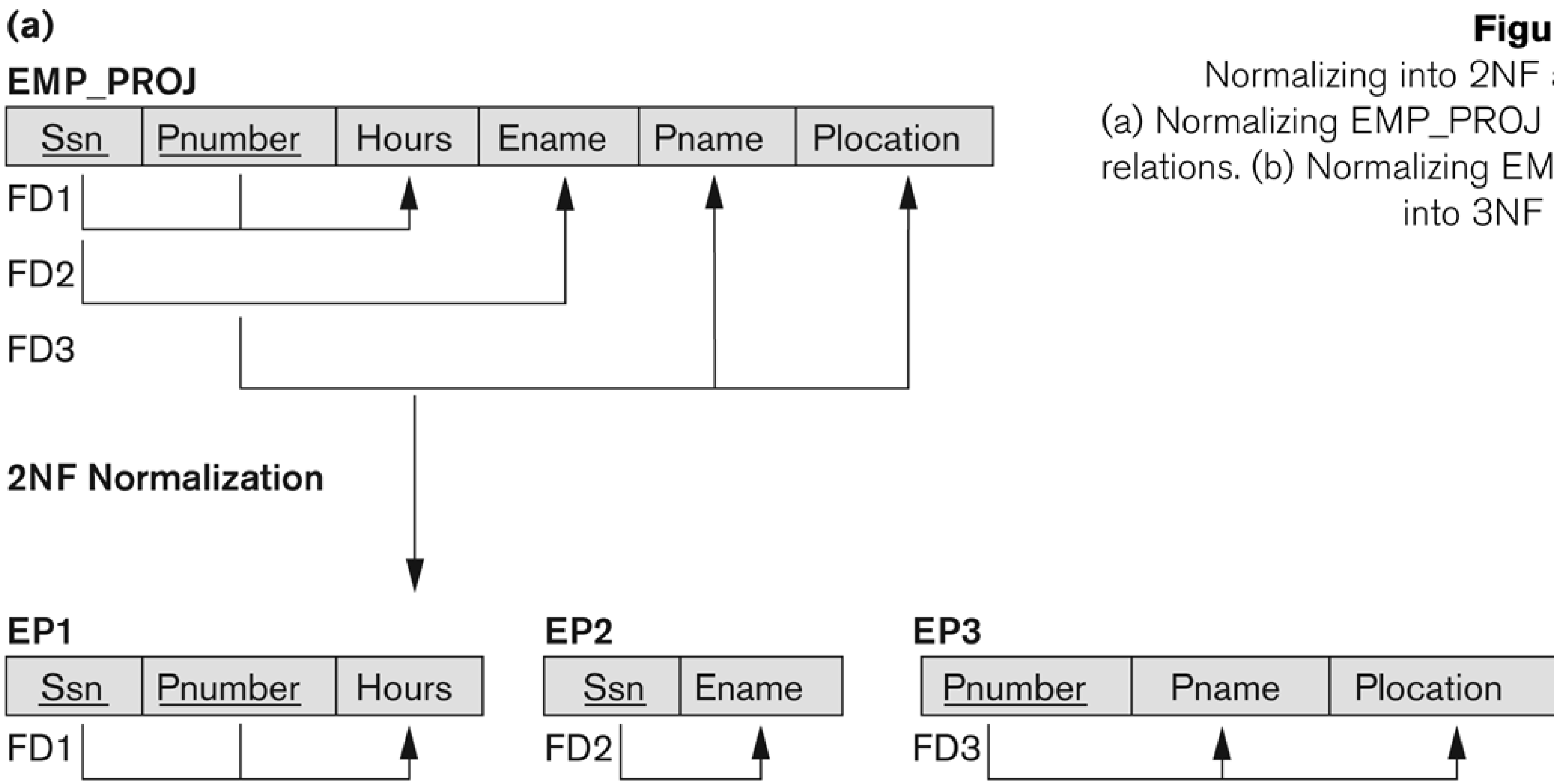
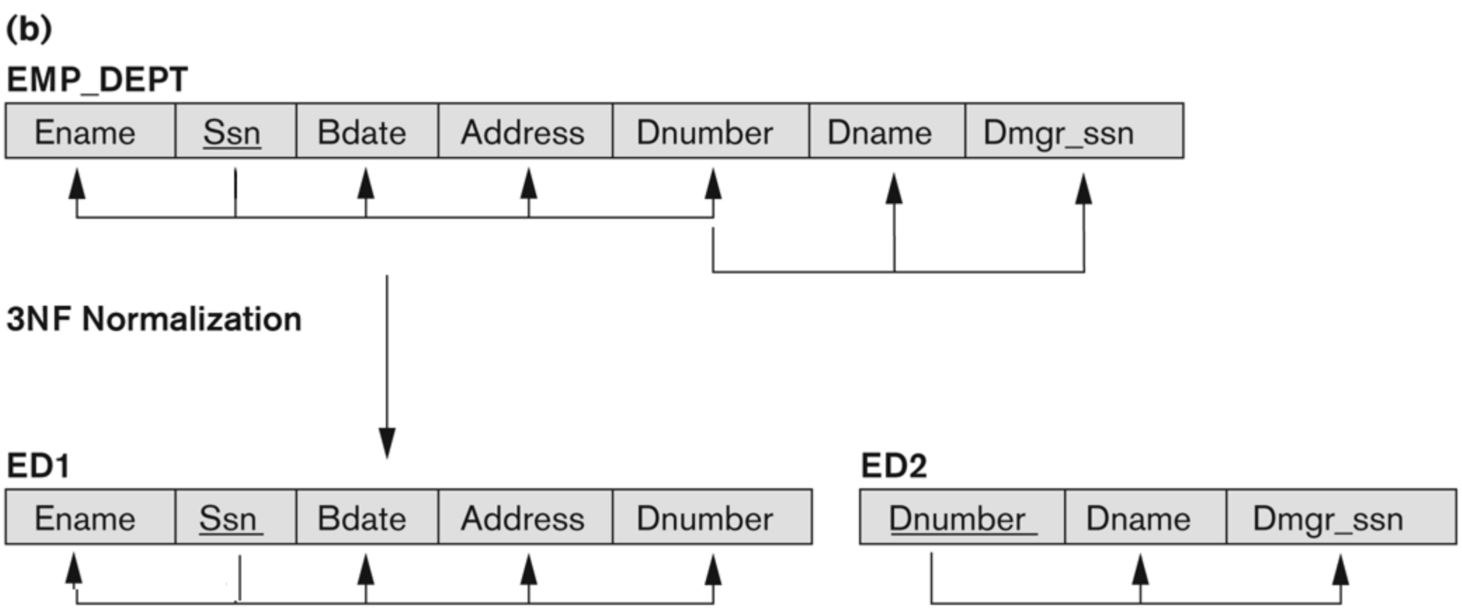


Figure
Normalizing into 2NF and 3NF.
(a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

Third Normal Form

- 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
  - **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


(b)
EMP_DEPT
Ename
Ssn
Bdate
Address
Dnumber
Dname
Dmgr_ssn
3NF Normalization
ED1
Ename
Ssn
Bdate
Address
Dnumber
ED2
Dnumber
Dname
Dmgr_ssn

Normal Forms Defined Informally

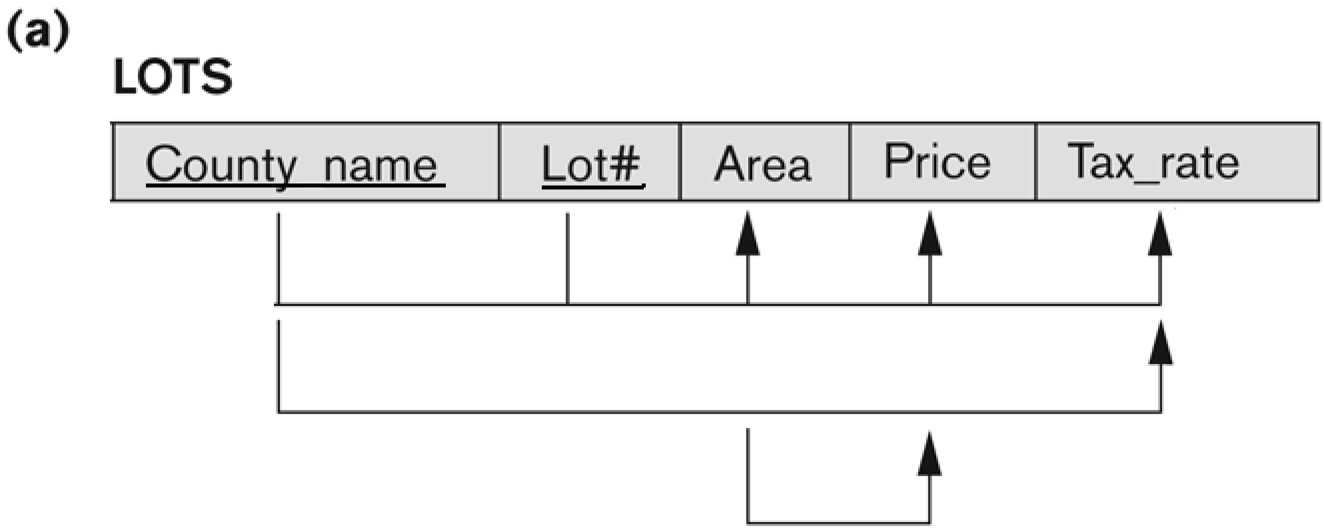
- 1st normal form
  - All attributes depend on **the key**
- 2nd normal form
  - All attributes depend on **the whole key**
- 3rd normal form
  - All attributes depend on **nothing but the key**

SUMMARY OF NORMAL FORMS based on Primary Keys

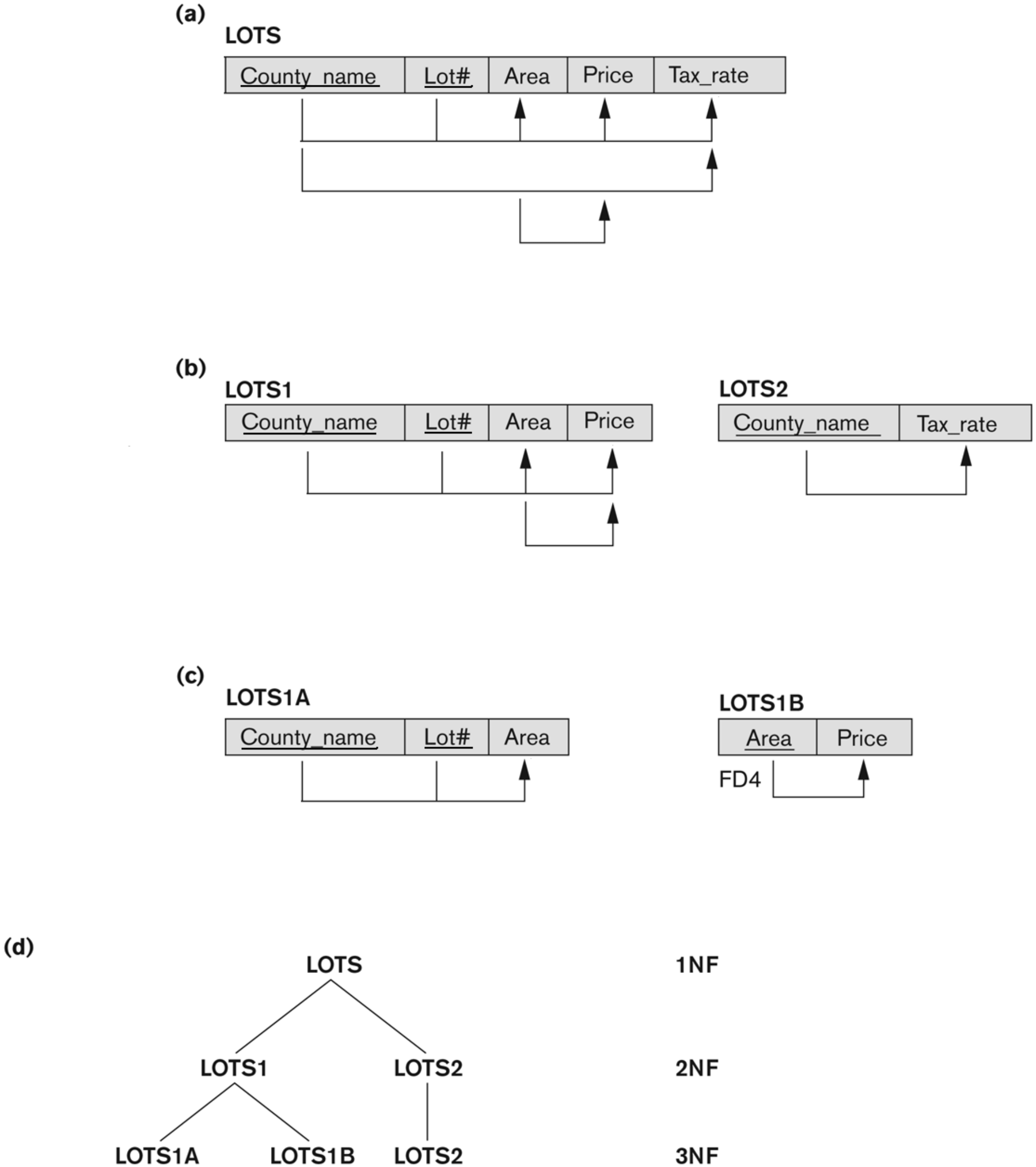
Table
Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

Practice



Find whether the above relation (fig (a)) is in
2NF, or 3NF? Why or why not? How would you successively normalize it completely?


(a)
LOTS
County_name
Lot#
Area
Price
Tax_rate
(b)
LOTS1
County_name
Lot#
Area
Price
LOTS2
County_name
Tax_rate
(c)
LOTS1A
County_name
Lot#
Area
LOTS1B
Area
Price
FD4
(d)
LOTS
1NF
LOTS1
LOTS2
2NF
LOTS1A
LOTS1B
LOTS2
3NF