Chapter 19: Normalization

- Refining database design
  - Main idea: Informally, each tuple in a relation should represent one entity or relationship instance.
  - The main objective is to create an accurate representation of data, relationship between the data, and constraints on the data that is relevant.
  - identify suitable set of relations (table) by creating good table structure/**process of assigning attributes to entities**. The process is known as ***Normalization*.**
- Process for evaluating and correcting table structures to minimize/control data redundancies {reduces data anomalies}.
- Works through a series of stages called normal forms.

Normalization Overview

- Each relationship can be normalized into a specific form to avoid *anomalies.*
  - *Anomalies?*
    - Anomaly = abnormality
    - Ideally a field value change, should be made only in a single place.
    - Data redundancy, promotes an abnormal condition by forcing field value changes in many different locations.

Redundancy

- Information is stored redundantly
  - Wastes storage
  - Causes problems with update anomalies
    - Insertion anomalies
    - Deletion anomalies
    - Modification anomalies

ssn	*name*	*lot*	*rating*	*hourly_wages*	*hours_worked*
123-22-3666	Attishoo	48	8	10	40
231-31-5368	Smiley	22	8	10	30
131-24-3650	Smethurst	35	5	7	30
434-26-3751	Guldu	35	5	7	32
612-67-4134	Madayan	35	8	10	40

Solutions?

- Decomposition
- Do we need to decompose the relation?
  - *Several normal forms have been found*

- What problems are associated with decomposition?

Design

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

Functional Dependencies

- A **functional dependency** (FD) is a constraint that generalizes the concept of a *key*

ssn	*name*	*lot*	*rating*	*hourly*
123-22-3666	Attishoo	48	8	10
231-31-5368	Smiley	22	8	10
131-24-3650	Smethurst	35	5	7
434-26-3751	Guldu	35	5	7
612-67-4134	Madayan	35	8	10

- Let *R* be a relation schema and let *X* nonempty sets of attributes in *R*. We

say that an instance *r* of *R* satisfies the FD $X \rightarrow Y$ if the following holds for

Every pair of tuples $t1$ and $t2$ in r:

<u>If $t1{:}X = t2{:}X$, then $t1{:}Y = t2{:}Y$</u> .

Every pair of tuples $t1$ and $t2$ in r:
<u>If $t1{:}X = t2{:}X$, then $t1{:}Y = t2{:}Y$</u> .

Example

- In the following relation: Do we have any FDs?

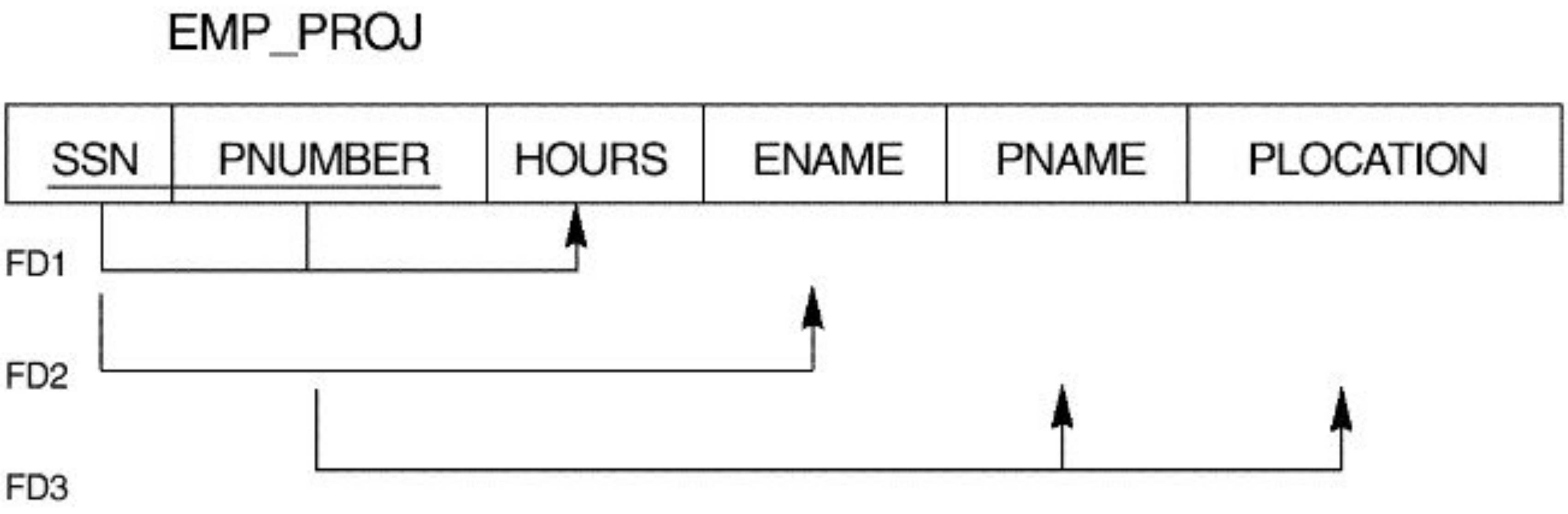
A	B	C	D
a1	b1	c1	d1
a1	b1	c1	d2
a1	b2	c2	d1
a2	b1	c3	d1

Keys and FDs

- A primary key constraint is a special case of an FD.
  - Note, however, that the definition of an FD does not require that the set *X* be minimal;
  - The additional minimality condition must be met for *X* to be a key
- Super Key
  - If $X \rightarrow Y$ holds, where *Y* is the set of all attributes, and there is some subset *V* of *X* such that $V \rightarrow Y$ holds, then *X* is a *superkey*;

- Keys

Consider the relation schema EMP_PROJ in from the semantics of the attributes, we know that the following functional dependencies should hold:
SSN □ ENAME
PNUMBER □ {PNAME, PLOCATION}
{SSN, PNUMBER} □ HOURS



Inference Rules for Functional Dependencies

- We denote by *F* the set of functional dependencies that are specified on relation schema *R*

- We usually specify the FDs that are semantically obvious
- But there are other FDs that can be detucted

Armstrong's Axioms

- **Sound & Complete**
- **Reflexivity**:
  - Y is a subset of X If $X \supseteq Y$, then $X \rightarrow Y$.
- **Augmentation**:
  - if X □ Y, then XZ □ YZ for any Z
- **Transitivity** :
  - if X □ Y and Y□ Z then X □ Z

- **Additional Useful Rules**
  - **Union**: If X□ Y and X□ Z, then X□YZ
  - **Decomposition**: if X□YZ then X□Y and X□Z

Examples on Armstrong Axioms: Union

- Prove Union:
- $$\frac{\begin{array}{c}X \to Y\\ X \to Z\end{array}}{X \to YZ}$$

- Step 1: Augmentation X
$$\frac{X \to Y}{X \to YX}$$
- Step 2: Augmentation Y
$$\frac{X \to Z}{XY \to YZ}$$
- Step 3: Transitivity of 1 and 2
$$\frac{\begin{array}{c}X \to YX\\ XY \to YZ\end{array}}{X \to YZ}$$

Examples on Armstrong Axioms

- Prove decomposition:

$$\frac{X \to YZ}{X \to Z}$$

- Step 1: Reflexivity, $Z \subseteq YZ$

$$YZ \to Z$$

- Step 2: Transitivity

$$X \to YZ$$

$$\frac{YZ \to Z}{X \to Z}$$

Examples on Armstrong Axioms

- Relation: ABCDEFGHIJ
- 1. AB□E
- 2. AG□J
- 3. BE□I
- 4. E□G
- 5. GI□H
- Prove AB□GH

- AB□ G
- AB□H
- Using union rule, AB-->GH

- Hint:
  - Start with the solution
  - Trace back
  - GH goes back to G and H
  - G goes back to E
  - E goes back AB
  - H goes back to GI
  - GI goes back to G and I
  - G goes back to E
  - E goes back to AB
  - I goes back to BE
  - BE goes back to AB

they imply about the relationship that the relation models).

Exercise 19.2 Consider a relation R with five attributes $ABCDE$. You are given the following dependencies: $A \rightarrow B$, $BC \rightarrow E$, and $ED \rightarrow A$.

1. List all keys for R.

they imply about the relationship that the relation models).

Exercise 19.2 Consider a relation R with five attributes $ABCDE$. You are given the following dependencies: $A \rightarrow B$, $BC \rightarrow E$, and $ED \rightarrow A$.

1. List all keys for R.

Normalization

- Takes a relation schema through a series of tests to "certify" whether it satisfies a certain **normal form**
- We have 4 normal forms
- All these normal forms are based on the functional dependencies among the attributes of a relation
- Unsatisfactory relation schemas that do not meet certain conditions—the **normal form tests**—are decomposed into smaller relation schemas that meet the tests and hence possess the desirable properties

First Normal Form (1NF)

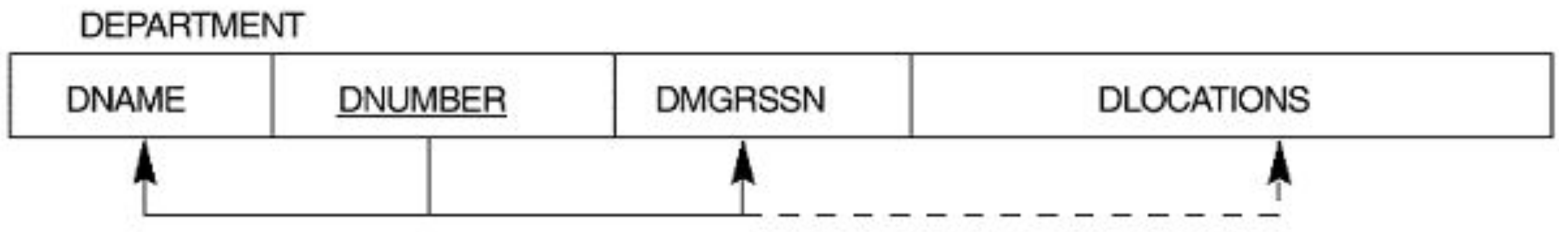
- Historically, it was defined to disallow multivalued attributes, composite attributes, and their combinations

- 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 that attribute.

- Already… all ours designs fulfil first normal form

Consider the DEPARTMENT relation schema shown

(a)



(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

Solutions?

- Remove the attribute DLOCATIONS that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key DNUMBER of DEPARTMENT

  --The primary key of this relation is the combination {DNUMBER, DLOCATION}
- Expand the key , In this case, the primary key becomes the combination {DNUMBER, DLOCATION}. This solution has the disadvantage of introducing *redundancy* in the relation. As in the prevous diagram (c)
- If a *maximum number of values* is known for the attribute—for example, if it is known that *at most three locations* can exist for a department—replace the DLOCATIONS attribute by three atomic attributes: DLOCATION1, DLOCATION2, and DLOCATION3. This solution has the disadvantage of introducing *null values* if most departments have fewer than three locations.

1NF

- The first normal form also disallows multivalued attributes that are themselves composite.

- These are called **nested relations** because each tuple can have a relation *within it.*

- Take a look at the following diagram:

(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

2NF

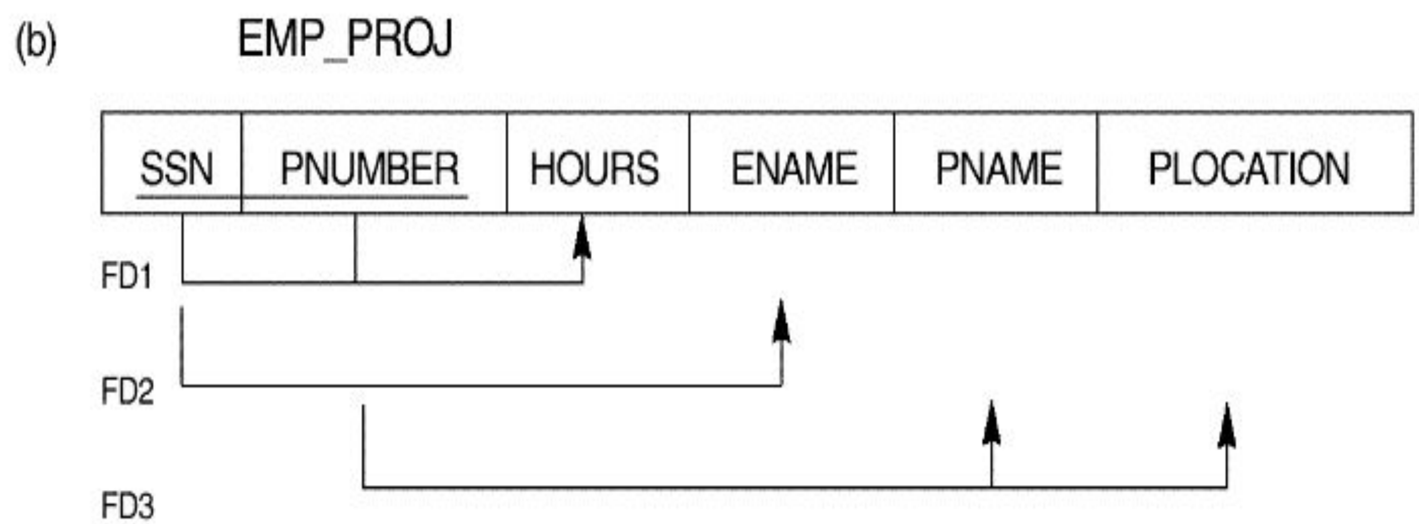
- **Second normal form (2NF)** is based on the concept of *full functional dependency.*

- A functional dependency $X \rightarrow Y$ is a **full functional dependency** if removal of any attribute A from X means that the dependency does not hold any more;

Partial Dependencies

Hours, fully FD on the key
Ename?? NO
SSN→ENAME
PNAME , Plocaiton NO

- A functional dependency $X \rightarrow Y$ is a **partial dependency** if some attribute $A \in X$ can be removed from X and the dependency still holds



For this to be in 2nd Normal Form,
Hours, ename and pname and plocation .. Individually have to be fully dependent on the key (ssn, pnumber)

Full Dependencies

- {SSN, PNUMBER}□ HOURS is a full dependency (neither SSN □ HOURS nor PNUMBER □ HOURS holds).

- However, the dependency {SSN, PNUMBER} □ ENAME is partial because SSN □ ENAME holds.

2NF

- The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key.

- If the primary key contains a single attribute, the test need not be applied at all.

non-key

- A relation schema *R* is in **2NF** if every nonprime attribute *A* in *R* is *fully functionally dependent* on the all keys of *R*

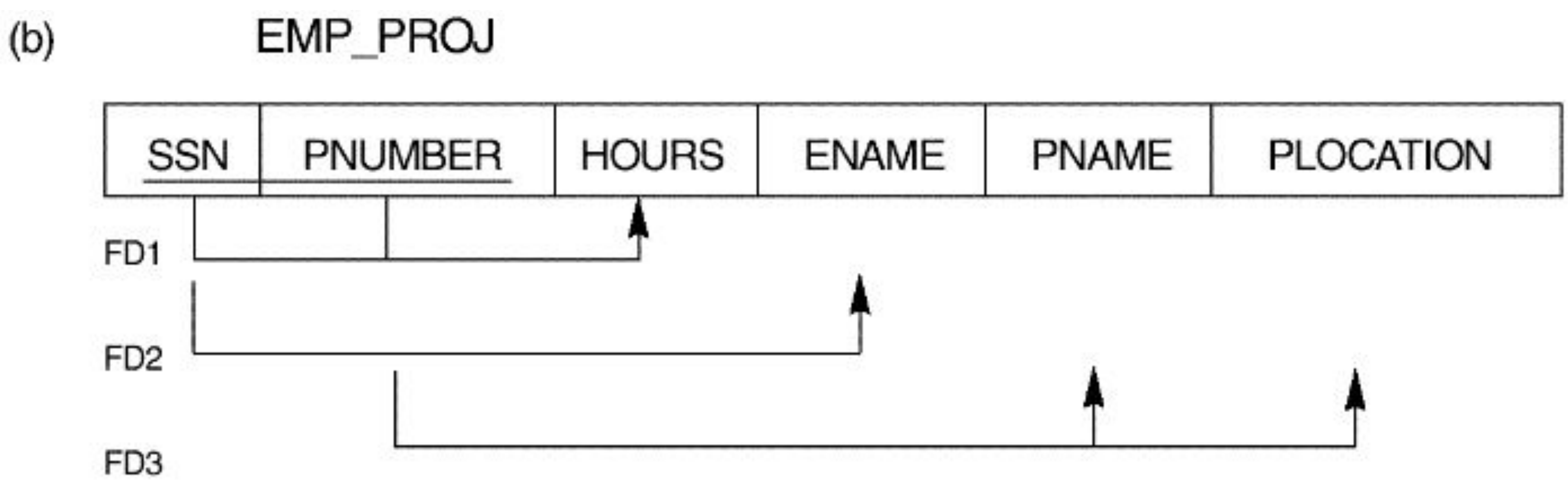
- All attributes (other than key attributes) have to be fully dependant on the key

3NF: Two equivalent Definitions

- According to Codd's original definition, a relation schema *R* is in **3NF** if it satisfies 2NF *and* no nonprime attribute of *R* is transitively dependent on all keys.
- A functional dependency $X \rightarrow Y$ in a relation schema *R* is a **transitive dependency** if there is a set of attributes *Z* that is neither a candidate key nor a subset of any key of *R* , and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold.

- Alternatively A relation is in 3NF if for all FDs $X \rightarrow A$ one of the following conditions must hold
  - $A \in X$ i.e. trivial FD or
  - X is a superkey (or key) or
  - A is part of some key

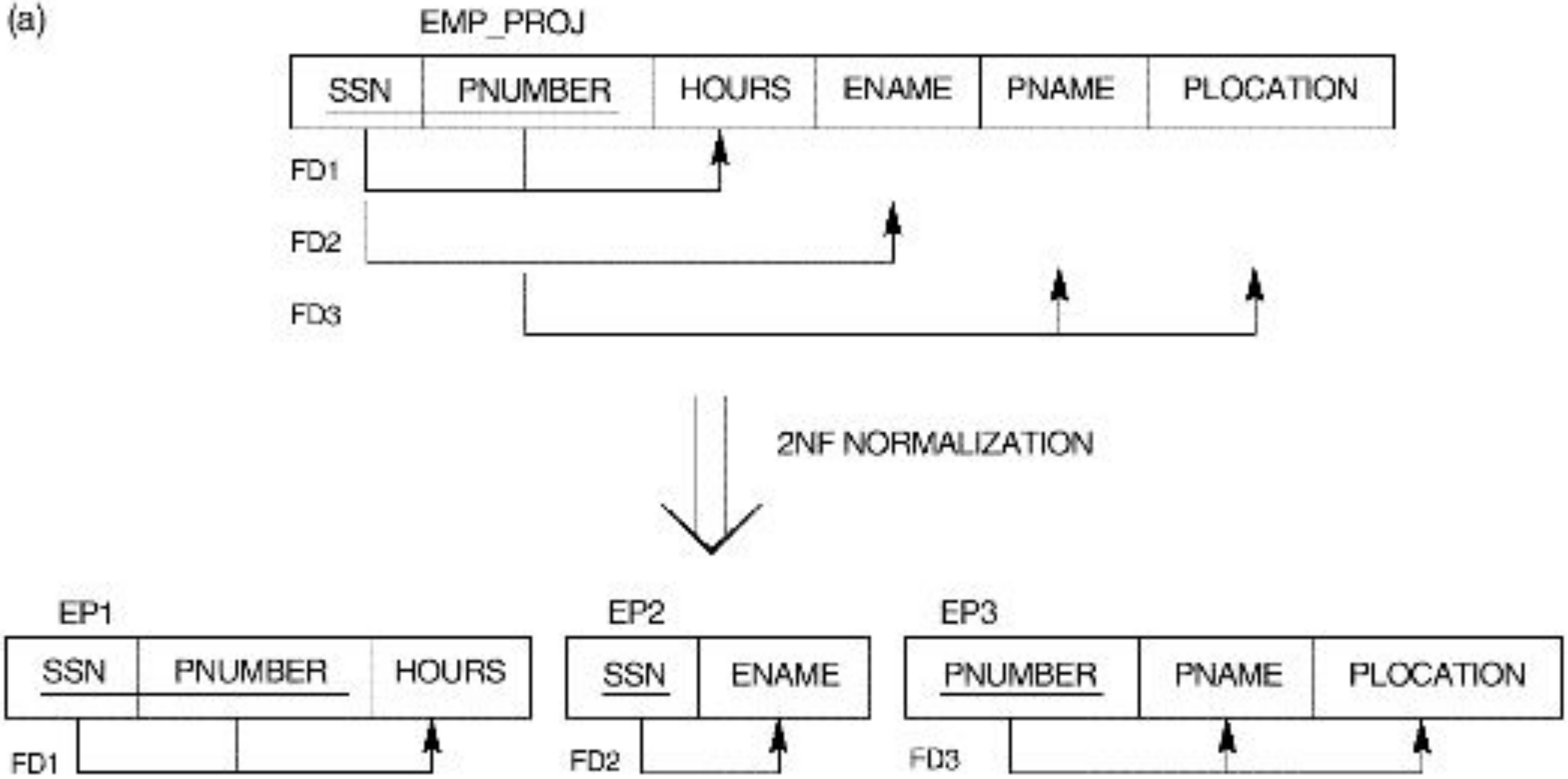
- The EMP_PROJ relation is in 1NF but is not in 2NF.
- The nonprime attribute ENAME violates 2NF . because of FD2, as do the nonprime . attributes PNAME and PLOCATION because of FD3



2NF Normalization

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

- The functional dependencies FD1, FD2, and FD3 in hence lead to the decomposition of EMP_PROJ into the three relation schemas EP1, EP2, and EP3 shown in


(a)
EMP_PROJ
SSN
PNUMBER
HOURS
ENAME
PNAME
PLOCATION
FD1
FD2
FD3
2NF NORMALIZATION
EP1
SSN
PNUMBER
HOURS
FD1
EP2
SSN
ENAME
FD2
EP3
PNUMBER
PNAME
PLOCATION
FD3

- As a general definition of **prime attribute,** an attribute that is part of *any candidate key* will be considered as prime.

- Partial and full functional dependencies and transitive dependencies will now be *with respect to all candidate keys* of a relation.

3NF

- According to Codd's original definition, a relation schema *R* is in **3NF** if it satisfies 2NF *and* no nonprime attribute of *R* is transitively dependent on all keys.
- Alternatively A relation is in 3NF if for all FDs X→A one of the following conditions must hold
  - $A \in X$ i.e. trivial FD or
  - X is a superkey (or key) or
  - A is part of some key

- The relation schema EMP_DEPT in is in 2NF, since no partial dependencies on a key exist.
- However, EMP_DEPT is not in 3NF
  - For FD Dnumber → Dname Dnumber is not a superkey

  (because of the transitive dependency of DMGRSSN (and also DNAME) on SSN via DNUMBER.)

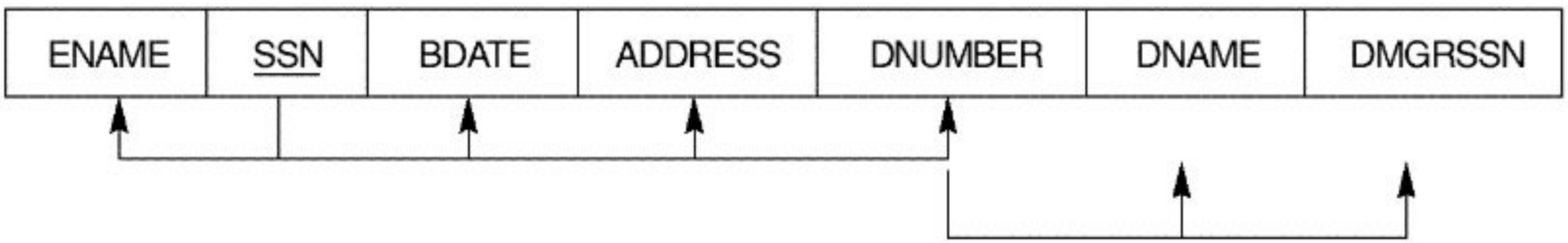
Example

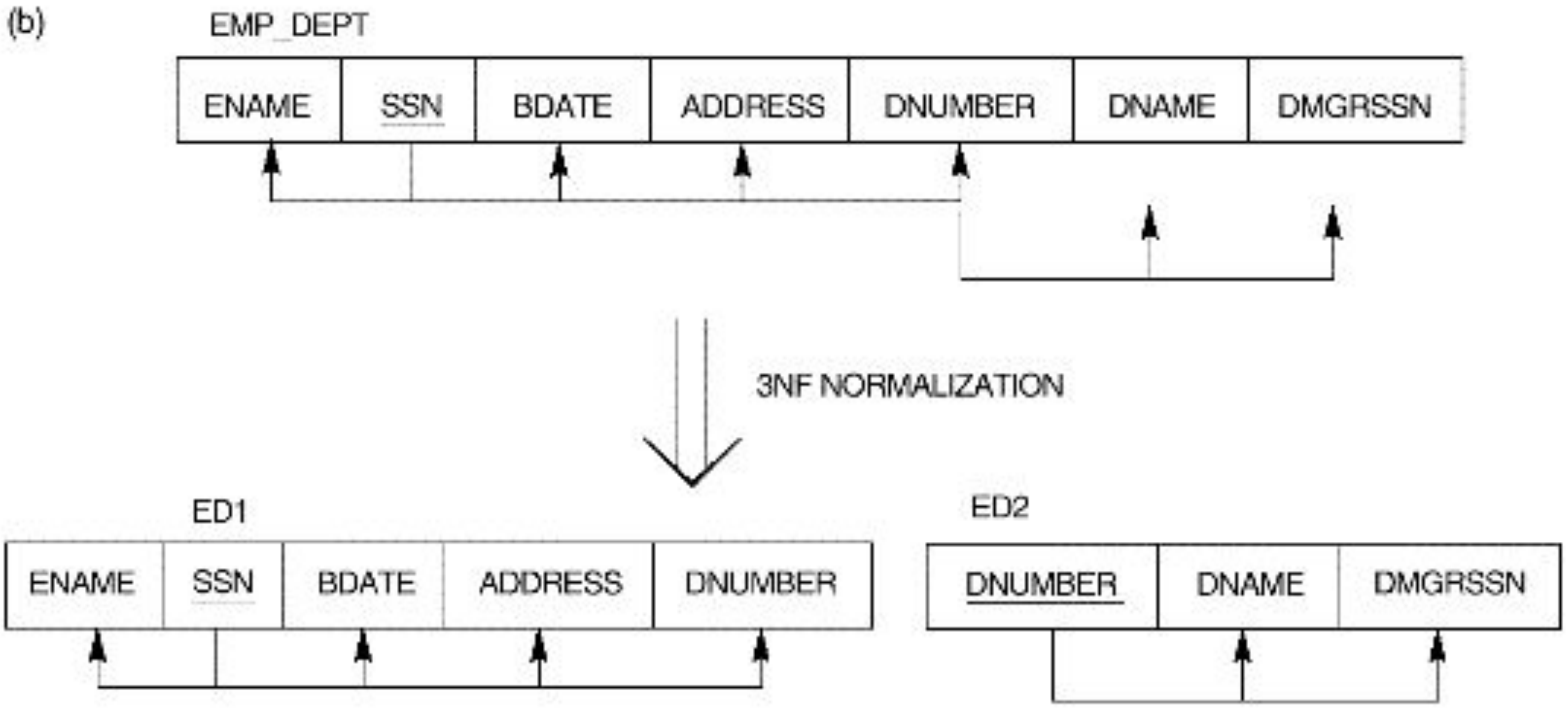
3NF:

for all FDs X→A one of the following conditions must hold

- $A \in X$ i.e. trivial FD or
- X is a superkey (or key) or
- A is part of some key

EMP_DEPT




(b)
EMP_DEPT
ENAME
SSN
BDATE
ADDRESS
DNUMBER
DNAME
DMGRSSN
3NF NORMALIZATION
ED1
ENAME
SSN
BDATE
ADDRESS
DNUMBER
ED2
DNUMBER
DNAME
DMGRSSN

- **Boyce-Codd normal form (BCNF)** was proposed as a simpler form of 3NF,

- but it was found to be stricter than 3NF,

- because every relation in BCNF is also in 3NF; however, a relation in 3NF is *not necessarily* in BCNF

BCNF

- Definition: <u>**A relation schema *R* is in BCNF if whenever a *nontrivial* functional dependency *X* □ *A* holds in *R*, then *X* is a superkey (candidate key)of *R*.**</u>

BCNF
for all FDs X→A one of the following conditions must hold
    $A \in X$ i.e. trivial FD or
    X is a superkey (or key)

3NF
for all FDs X→A one of the following conditions must hold
    $A \in X$ i.e. trivial FD or
    X is a superkey (or key) or
    A is part of some key

- In our example, AREA→ County_name violates BCNF in LOTS1A because AREA is not a superkey of LOTS1A

- Note that FD5 satisfies 3NF in LOTS1A because COUNTY_NAME is a prime attribute

for all FDs X→A one of the following conditions must hold

$A \in X$ i.e. trivial FD or

X is a superkey (or key) or

A is part of some key

Consider the universal relation R = {A, B, C, D, E, F, G, H, I, J} and the set of functional dependencies F =

A, B□ C,

A □ D, E,

B □ F,

F □ G, H,

D □I, J.

What is the key for R

Consider a relation R(A, B, C, D, E) with the following dependencies:

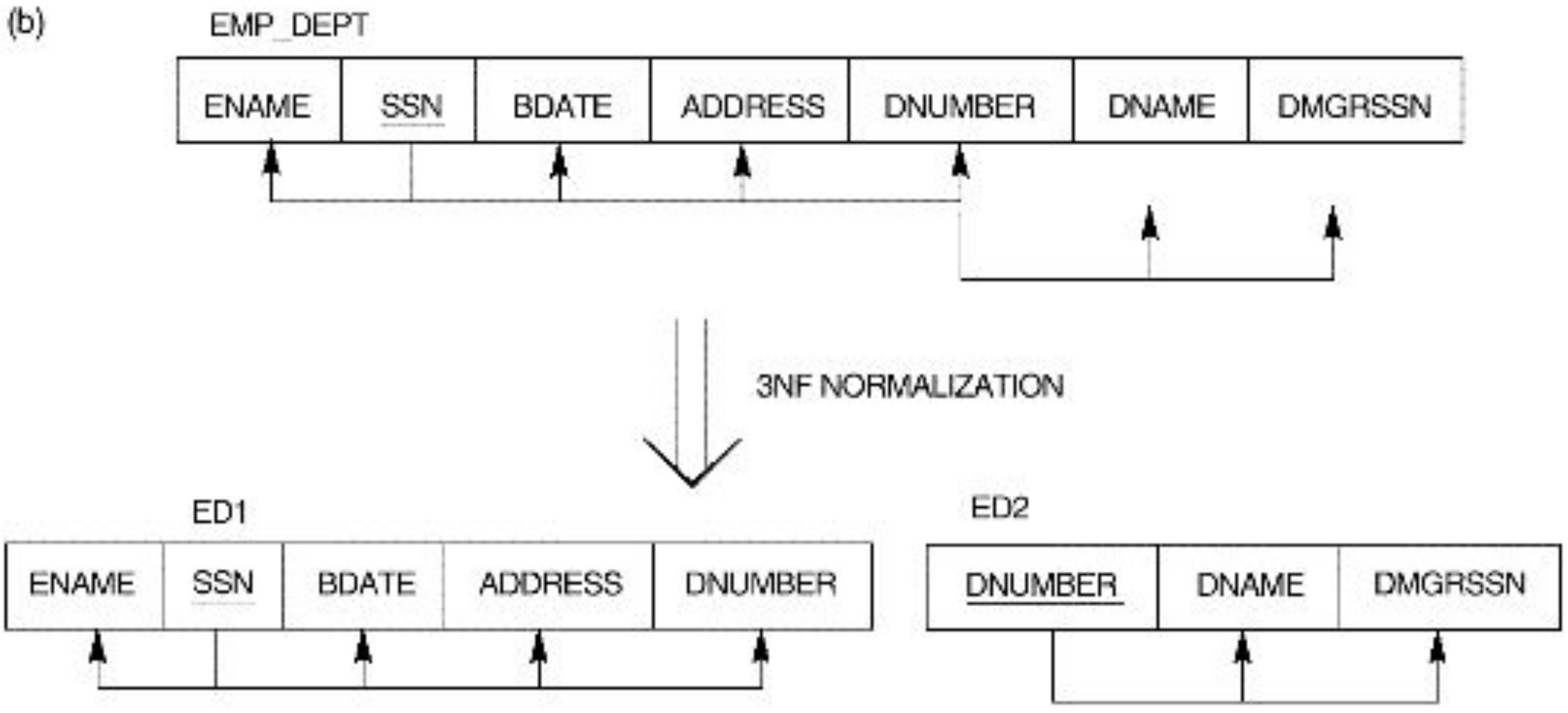
AB → C, CD → E, DE → B

Exercise 19.7 Suppose you are given a relation R with four attributes $ABCD$. For each of the following sets of FDs, assuming those are the only dependencies that hold for R, do the following: (a) Identify the candidate key(s) for R. (b) Identify the best normal form that R satisfies (1NF, 2NF, 3NF, or BCNF). (c) If R is not in BCNF, decompose it into a set of BCNF relations that preserve the dependencies.

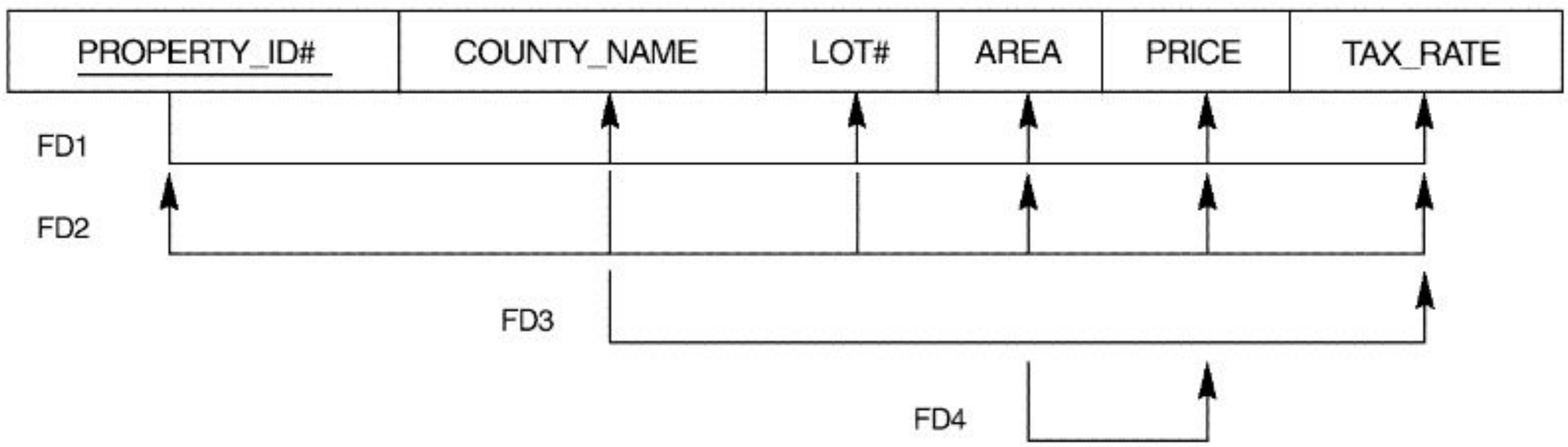
1. $C \rightarrow D,\ C \rightarrow A,\ B \rightarrow C$
2. $B \rightarrow C,\ D \rightarrow A$
3. $ABC \rightarrow D,\ D \rightarrow A$
4. $A \rightarrow B,\ BC \rightarrow D,\ A \rightarrow C$
5. $AB \rightarrow C,\ AB \rightarrow D,\ C \rightarrow A,\ D \rightarrow B$

3NF
for all FDs X→A one of the following conditions must hold

- $A \in X$ i.e. trivial FD or
- X is a superkey (or key) or
- A is part of some key


(b)
EMP_DEPT
ENAME
SSN
BDATE
ADDRESS
DNUMBER
DNAME
DMGRSSN
3NF NORMALIZATION
ED1
ENAME
SSN
BDATE
ADDRESS
DNUMBER
ED2
DNUMBER
DNAME
DMGRSSN

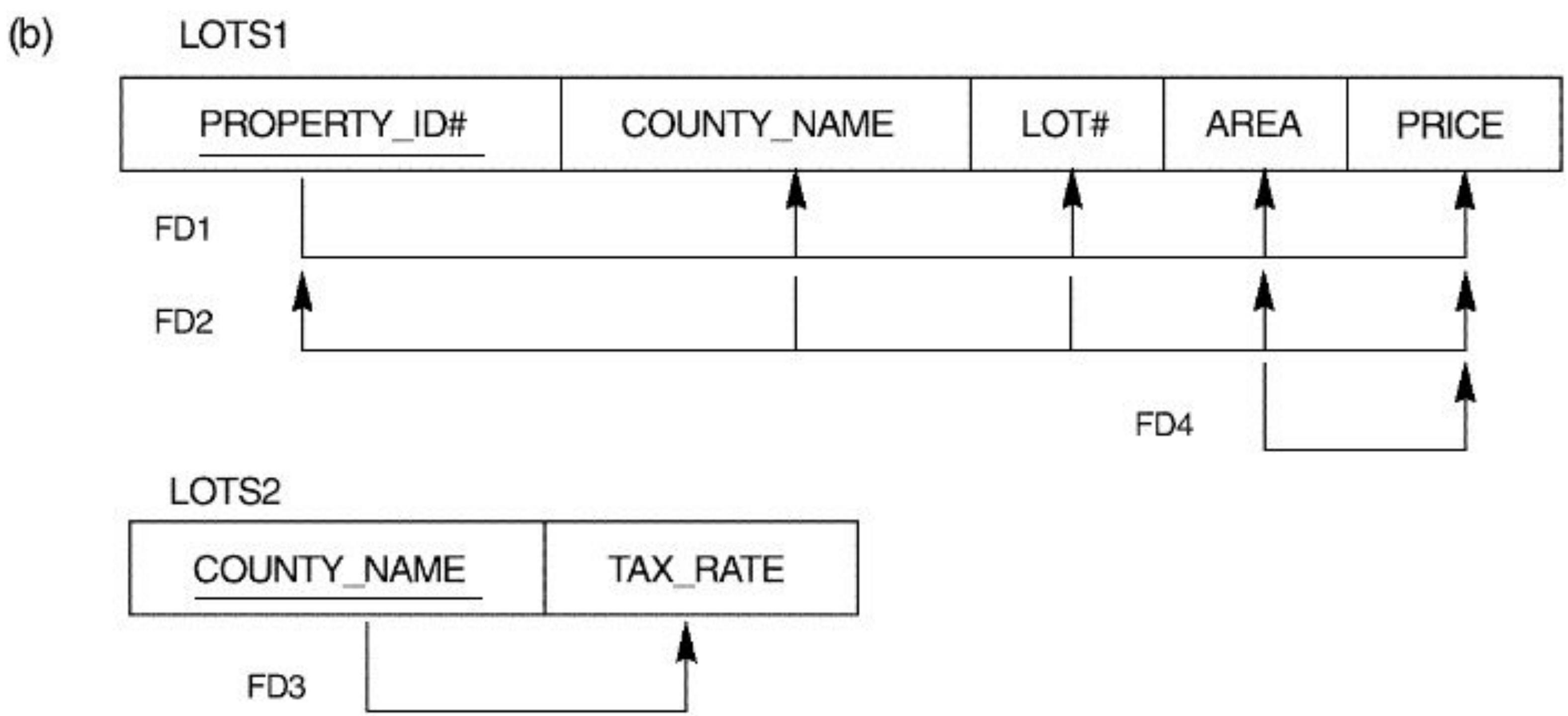
(a) LOTS


PROPERTY_ID#
COUNTY_NAME
LOT#
AREA
PRICE
TAX_RATE
FD1
FD2
FD3
FD4

- The LOTS relation schema violates the general definition of 2NF

- because TAX_RATE is partially dependent on the candidate key {COUNTY_NAME, LOT#}, due to FD3

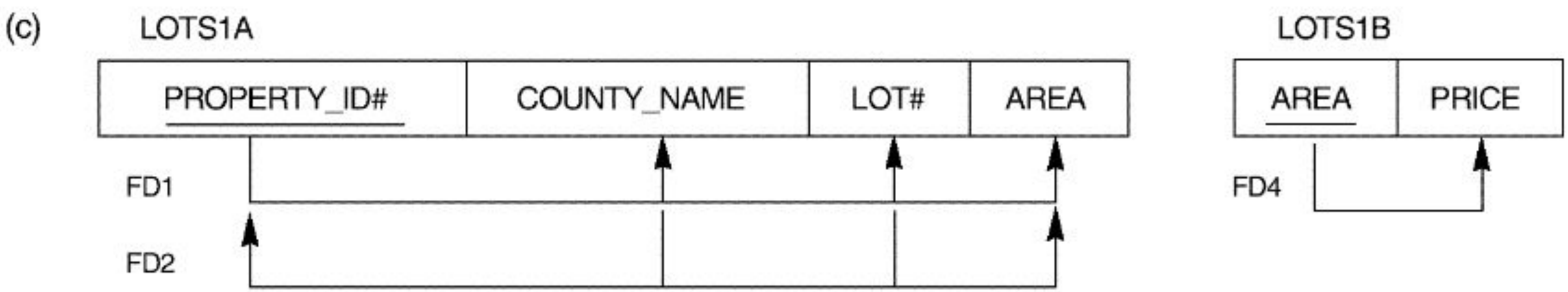
- To normalize LOTS into 2NF, we decompose it into the two relations LOTS1 and LOTS2,



LOTS2 is in 3NF. However, FD4 in LOTS1 violates 3NF because AREA is not a candidate key and PRICE is not a prime attribute in LOTS1

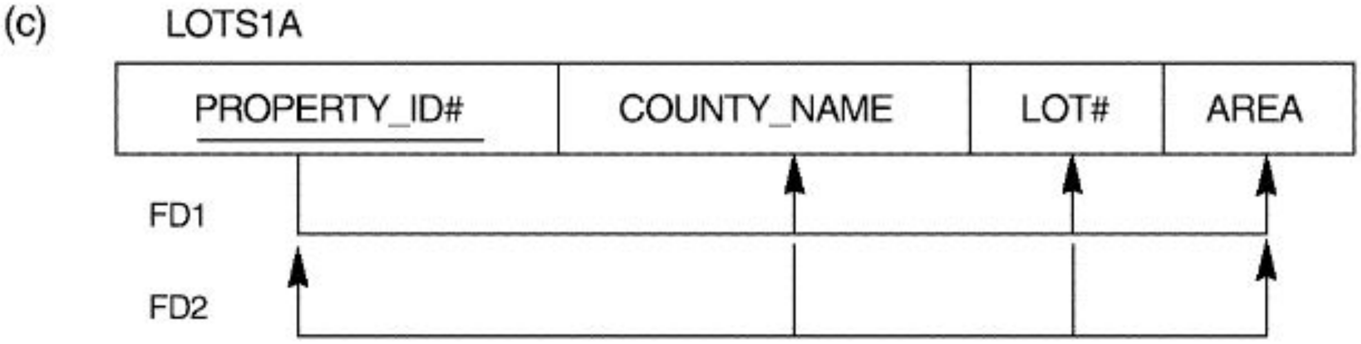
- LOTS2 is in 3NF. However, FD4 in LOTS1 violates 3NF because AREA is not a candidate key and PRICE is not a prime attribute in LOTS1

- To normalize LOTS1 into 3NF, we decompose it into the relation schemas LOTS1A and LOTS1B



- **Boyce-Codd normal form (BCNF)** was proposed as a simpler form of 3NF,
- but it was found to be stricter than 3NF,
- because every relation in BCNF is also in 3NF; however, a relation in 3NF is *not necessarily* in BCNF

- Lets go back to this schema
- the relation schema LOTS1A still is in 3NF because COUNTY_NAME is a prime attribute.



And let us add this FD AREA County_Name

BCNF

- Definition: <u>**A relation schema *R* is in BCNF if whenever a *nontrivial* functional dependency *X* □ *A* holds in *R*, then *X* is a superkey (candidate key)of *R*.**</u>

- In our example, AREA County_name violates BCNF in LOTS1A because AREA is not a superkey of LOTS1A

- Note that FD5 satisfies 3NF in LOTS1A because COUNTY_NAME is a prime attribute

BCNF
for all FDs X→A one of the following conditions must hold
$A \in X$ i.e. trivial FD or
X is a superkey (or key)

3NF
for all FDs X→A one of the following conditions must hold
$A \in X$ i.e. trivial FD or
X is a superkey (or key) or
A is part of some key

- We can decompose LOTS1A into two BCNF relations LOTS1AX and LOTS1AY,