



**BITS Pilani**  
Pilani Campus

# Database Systems (CSF212) Lecture – 17



**BITS Pilani**  
Pilani Campus



# Relational Database Design

# Relational Database Design

- Features of Good Relational Design
- Atomic Domains and First Normal Form
- Decomposition Using Functional Dependencies
- Functional Dependency Theory
- Algorithms for Functional Dependencies
- Decomposition Using Multi-valued Dependencies
- More Normal Form
- Database-Design Process
- Modeling Temporal Data

# Features of Good Relational Designs

Undesirable properties that a bad database design may have:

- Repetition of information
- Inability to represent certain information
- Loss of information

**branch**(branch-name, assets, branch-city)

**borrow**(branch\_name, loan-number,  
customer-name, amount)

# Features of Good Relational Designs

## – Repetition of information

branch		
branch-name	assets	branch-city
Downtown	9000	Brooklyn
Redwood	10000	Palo Alto
Perryridge	1000	Horseneck
Mianus	4000	Horseneck
Round Hill	80000	Horseneck
Brighton	7100	Brooklyn

borrow			
branch-name	LN	CN	AMT
Downtown	17	jones	1000
Redwood	23	Smith	2000
Perry ridge	15	Hayes	1500
Downtown	14	Jackson	1500
Mianus	93	Curry	500
Roundhill	11	Turner	900
Perryridge	25	Glenn	2500
Brighton	10	Brooks	2200

# Features of Good Relational Designs

- Repetition of information

branch-borrow					
branch-name	assets	branch-city	LN	CN	AMT
Downtown	9000	Brooklyn	17	jones	1000
Redwood	10000	Palo Alto	23	Smith	2000
Perryridge	1000	Horseneck	15	Hayes	1500
Mianus	4000	Horseneck	93	Curry	500
Round Hill	80000	Horseneck	11	Turner	900
Perryridge	1000	Horseneck	25	Glenn	2500
Brighton	7100	Brooklyn	10	Brooks	2200

# Features of Good Relational Designs

- Add a new loan (Perryridge, 31, Turner, 500) to both the designs

<b>borrow</b>			
BN	LN	CN	AMT
Downtown	17	jones	1000
Redwood	23	Smith	2000
Perry ridge	15	Hayes	1500
Downtown	14	Jackson	1500
Mianus	93	Curry	500
Roundhill	11	Turner	900
Perryridge	25	Glenn	2500
Brighton	10	Brooks	2200
<b>Perryridge</b>	<b>31</b>	<b>Turner</b>	<b>500</b>

# Features of Good Relational Designs

- Addition of new tuple leads to repetition of information

branch-borrow					
branch-name	assets	branch-city	LN	CN	AMT
Downtown	9000	Brooklyn	17	jones	1000
Redwood	10000	Palo Alto	23	Smith	2000
Perryridge	1000	Horseneck	15	Hayes	1500
Mianus	4000	Horseneck	93	Curry	500
Round Hill	80000	Horseneck	11	Turner	900
Perryridge	1000	Horseneck	25	Glenn	2500
Brighton	7100	Brooklyn	10	Brooks	2200
Perryridg e	1000	Horsenec k	31	Turner	500



# Features of Good Relational Designs

- Why repetition is undesirable?
  - Waste of space
  - Causes difficulty in updating a database
  - **Modification anomalies**
    - If Perryridge moves from Horseneck to Newtown
  - **Deletion anomalies**
    - If the details of the customer who happens to be the last borrower from a branch is deleted, then the information about the branch is also lost


# Features of Good Relational Designs

- Why repetition is undesirable?
- Certain information cannot be represented
  - Cannot store information about a branch if no loans exist (**insertion anomalies**)
  - Can use null values, but they are difficult to handle

# Features of Good Relational Designs

– Is having smaller schema a solution?

borrow			
BN	LN	CN	AMT
Downtown	17	jones	1000
Perry ridge	15	Hayes	1500
Downtown	14	Jackson	1500



loan		
BN	LN	AMT
Downtown	17	1000
Perry ridge	15	1500
Downtown	14	1500

amount	
CN	AMT
jones	1000
Hayes	1500
Jackson	1500

# Features of Good Relational Designs

– Is having smaller schema a

loan		
BN	LN	AMT
Downtown	17	1000
Perry ridge	15	1500
Downtown	14	1500

amount	
CN	AMT
Jones	1000
Hayes	1500
Jackson	1500

loan-amount			
BN	LN	CN	AMT
Downtown	17	jones	1000
Perry ridge	15	Hayes	1500
Perryridge	15	Jackson	1500
Downtown	14	Hayes	1500
Downtown	14	Jackson	1500

# Features of Good Relational Designs

- Although the resulting relation has more tuples, it contains less information
  - For example, we cannot find which customers are borrowers from which branch

$\Pi_{\text{BN}} (\sigma_{\text{CN}=\text{"Jackson"}} \text{ loan-amount})$

loan-amount			
BN	LN	CN	AMT
Perryridge	15	Jackson	1500
Downtown	14	Jackson	1500

← incorrect information

# Features of Good Relational Designs

- Such decompositions are called *lossy-decomposition* or *lossy-join decomposition*
- **Lossless-join decomposition:** A decomposition that is not a lossy-join decomposition.

# Example of Lossless-Join Decomposition

Decomposition of  $\mathbf{R} = (A, B, C)$  into  $\mathbf{R}_1 = (A, B)$  and  $\mathbf{R}_2 = (B, C)$

A	B	C
$\alpha$	1	A
$\beta$	2	B

$r$

A	B
$\alpha$	1
$\beta$	2

$\mathbf{r}_1 = \Pi_{A,B}(r)$

B	C
1	A
2	B

$\mathbf{r}_2 = \Pi_{B,C}(r)$

A	B	C
$\alpha$	1	A
$\beta$	2	B

$\Pi_{A,B}(r) \bowtie \Pi_{B,C}(r)$

***In general decomposition is lossless provided certain functional dependencies hold***

# Goal — Devise a Theory for the Following

- Decide whether a particular relation  $R$  is in “good” form.
- In the case that a relation  $R$  is not in “good” form, decompose it into a set of relations  $\{R_1, R_2, \dots, R_n\}$  such that
  - each relation is in good form
  - the decomposition is a lossless-join decomposition
- Our theory is based on:
  - functional dependencies
  - multi-valued dependencies



## **Summary of Features of Good Relational Designs**

- Free from various forms of anomalies
- Causes very less wastage of storage spaces due to NULLS
- Does not generate spurious data during joins

# Normalization

- Normalization: Process of analyzing a relational schema based on their **functional dependencies** and **primary keys** to achieve the desirable properties of
  - Minimizing redundancy
  - Minimizing anomalies
- Schemas that do not satisfy “normal form properties” are decomposed into smaller relational schemas that satisfies these properties
- NF of a relation: Refers to the highest NF condition that it meets

# Normalization

- **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

# Normalization

- Normalization through decomposition must also confirm the existence of following additional properties that the relational schema should possess:
  - Lossless join – Guarantees that the spurious tuple generation problem does not occur
  - Dependency preservation – Although not mandatory, each functional dependency should hold after decomposition

# First Normal Form (1NF)

- Domain is **atomic** if its elements are considered to be indivisible units
  - Examples of non-atomic domains:
    - Set of names, composite attributes
    - Identification numbers like CS101 that can be broken up into parts
- A relational schema R is in **first normal form** if the domains of all attributes of R are atomic
- Non-atomic values complicate storage and encourage redundant (repeated) storage of data
  - Example: Set of accounts stored with each customer, and set of owners stored with each account

# First Normal Form

customer		
CN	CID	C_ADDR
Jones	1	ABC street, Downtown
Jones	2	XYZ street, Perryridge



customer			
CN	CID	C_Street	C_City
Jones	1	ABC	Downtown
Jones	2	XYZ	Perryridge

New attributes are added

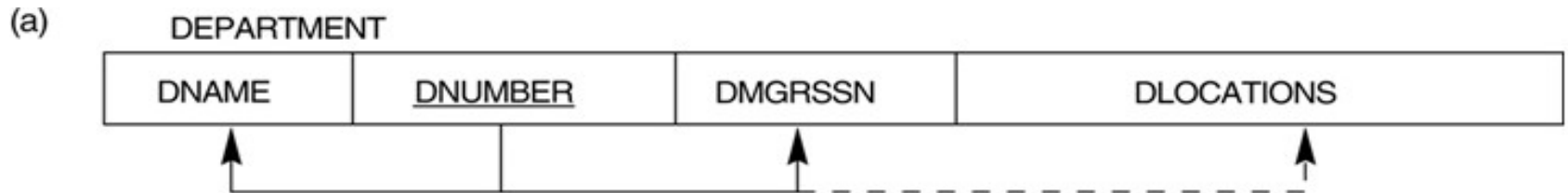
loan		
BN	LN	CID
Downtown	17, 20	1
Perry ridge	15	2



loan		
BN	LN	CN
Downtown	17	1
Downtown	20	1
Perryridge	15	2

New tuples are added

# First Normal Form



(b)

DEPARTMENT

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

(c)

DEPARTMENT

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

# Decomposition Using Functional Dependencies

- There are usually a variety of constraints (rules) on the data in the real world.
  - Students and instructors are uniquely identified by their ID
  - Each student and instructor has only one name, etc.
- *Legal instance of a relation*: An instance of a relation that satisfies all real-world constraints
- *Legal instance of a database*: Where all the relation instances are legal instances
- Constraints are represented formally as keys (super keys, candidate keys and primary keys), or as functional dependencies



# Functional Dependency

- 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

- Notation used for representing database elements:
  - **Greek letters**: For sets of attributes (for example,  $\alpha$ )
  - $r(R)$ : Schema is for relation  $r$ , where  $R$  denotes the set of attributes. Just  $R$  is used
  - $R$ : When the relation name does not matter
  - $K$ : A set of attributes that forms the Superkey
  - $t[\alpha]$ : Values in  $t$  on attributes in set  $\alpha$
  - **Lower case names**: For relations (for example, *instructor*). In Definitions and algorithms, single letters, like  $r$  is used

# Functional Dependencies

- Consider a relation schema  $r(R)$ , and let  $\alpha \subseteq R$  and  $\beta \subseteq R$ :
  - Given an instance of  $r(R)$ , we say that the instance *satisfies the functional dependency*  $\alpha \rightarrow \beta$ , if for all pairs of tuples  $t_1, t_2 \in r$ , if  $t_1[\alpha] = t_2[\alpha] \Rightarrow t_1[\beta] = t_2[\beta]$
  - We say that the functional dependency  $\alpha \rightarrow \beta$  *holds* on schema  $r(R)$  if, in every legal instance of  $r(R)$  it satisfies the functional dependency
- Example: Consider  $r(A,B)$  with the following instance of  $r$ :

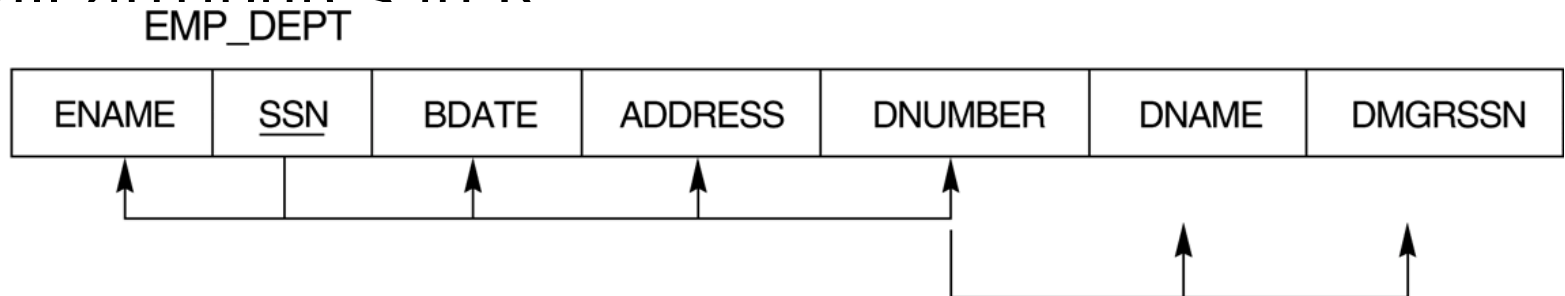
A	B
1	4
1	5
2	7

On this instance,  $A \rightarrow B$  does **NOT** hold, but  $B \rightarrow A$  does hold. So, we can say  $B$  functionally determines  $A$

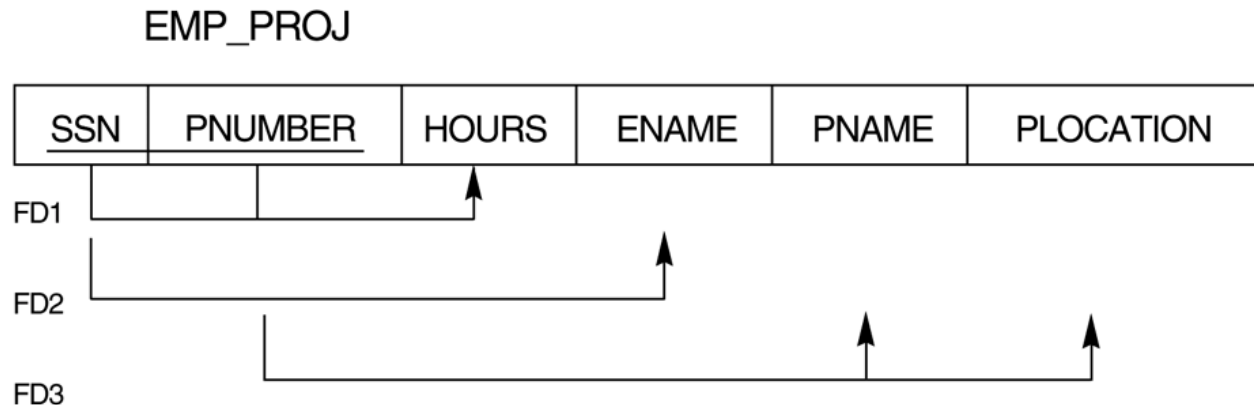
# Functional Dependencies

- 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

(a)



(b)

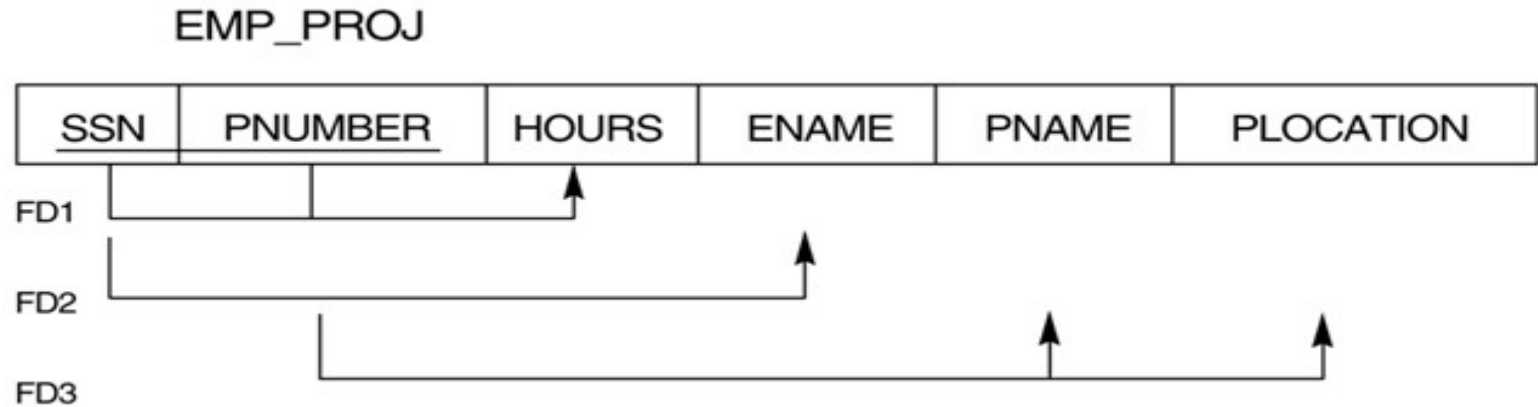


# Use of Functional Dependencies

- Specify constraints on the set of legal relations
- To define keys of a database
  - $K$  is a superkey for relation schema  $R$  if and only if  $K \rightarrow R$
  - $K$  is a candidate key for  $R$  if and only if
    - $K \rightarrow R$
    - For no  $\alpha \subset K$ ,  $\alpha \rightarrow R$

# Functional Dependencies

- A functional dependency  $\alpha \rightarrow \beta$  is **trivial** if  $\beta \subseteq \alpha$



## Trivial FD's:

$\text{SSN, PNUMBER} \rightarrow \text{SSN, PNUMBER}$

$\text{SSN, PNUMBER} \rightarrow \text{PNUMBER}$

$\text{SSN, PNUMBER} \rightarrow \text{SSN}$

$\text{SSN} \rightarrow \text{SSN}$

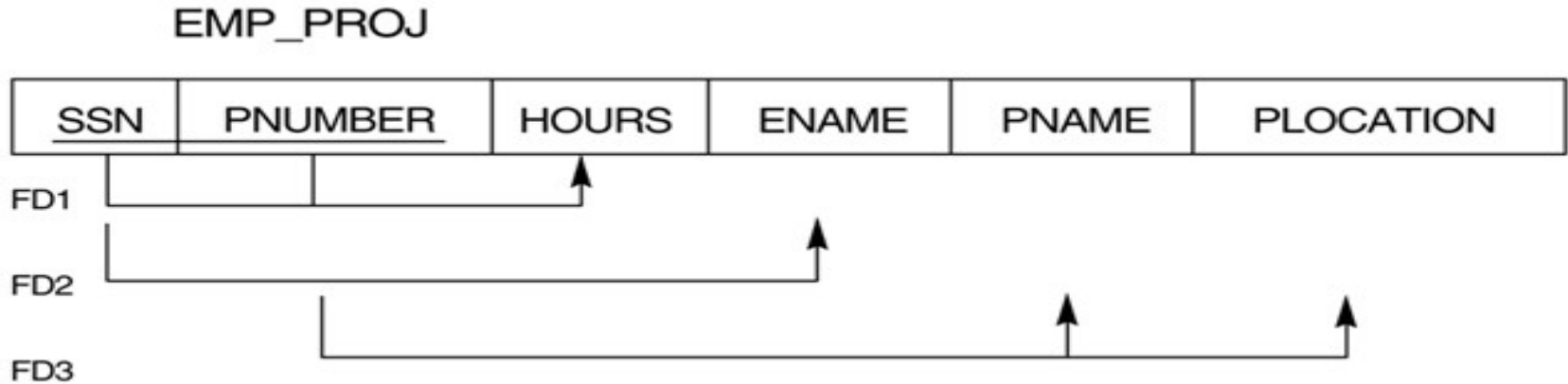
$\text{PNUMBER} \rightarrow \text{PNUMBER}$

# Definitions

Uses the concepts of **Partial FDs** and **Candidate key**

- **Prime attribute** - Attribute that is member of the candidate key. Other attributes are called as **non-prime key**
- **Full functional dependency** - A FD  $Y \rightarrow Z$  where removal of any attribute from Y means the FD does not hold any more
- **Partial dependency:** Partial Dependency occurs when a non-prime attribute is functionally dependent on part of a candidate key
  - They introduce redundancy in the relation

# Example



SSN AND PNUMBER ARE PK SET, SO THEY CAN DETERMINE ENAME. HOWEVER WE ALREADY HAVE SSN->ENAME AS A FD, HENCE SSN,PNUMBER->ENAME IS A PARTIAL DEP.

## Full dependencies

$\{\text{SSN}, \text{PNUMBER}\} \rightarrow \text{HOURS}$

## Partial dependencies

$\{\text{SSN}, \text{PNUMBER}\} \rightarrow \text{ENAME}$   
 $\{\text{SSN}, \text{PNUMBER}\} \rightarrow \text{PNAME}$   
 $\{\text{SSN}, \text{PNUMBER}\} \rightarrow \text{PLOCATION}$

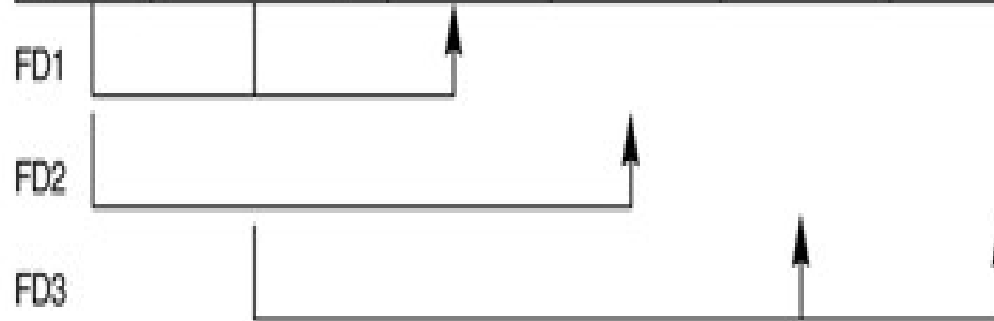


# Second Normal Form

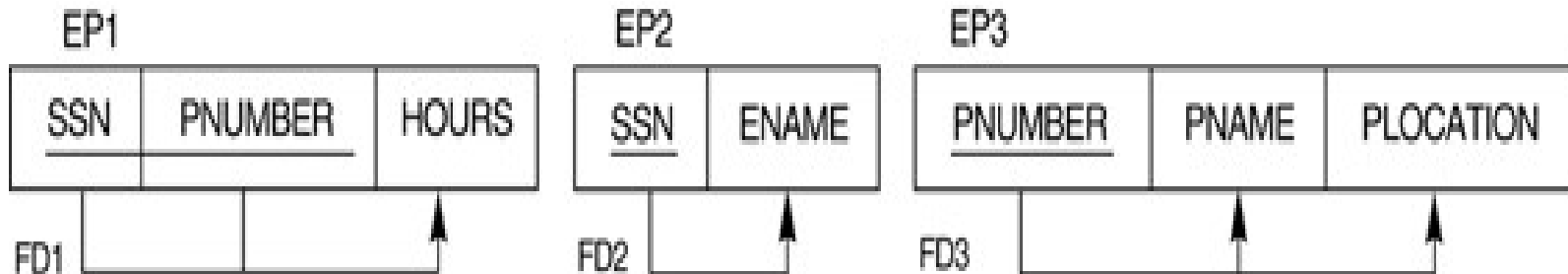
- A relation schema R is in **second normal form (2NF)** if:
  - It is in 1NF
  - Every non-prime attribute A in R is fully functionally dependent on the candidate key
- **2NF Decomposition:**
  - Remove all non prime attributes which are partially dependent on the primary key from the original relation
  - Create a new relation in which non prime attributes are associated only with the part of the primary key on which they are fully functionally dependent

# EMP\_PROJ

<u>SSN</u>	<u>PNUMBER</u>	HOURS	ENAME	PNAME	PLOCATION
------------	----------------	-------	-------	-------	-----------



2NF NORMALIZATION



# Need for 2NF

EMP_PROJ					
<u>SSN</u>	<u>PNUMBER</u>	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

# Second Normal Form

- **banker**(c\_id, agent\_no, br\_code)

Constraints:

- A agent can work for only one branch
- A agent can serve many customers

agent\_no  $\rightarrow$  br\_code

br\_code, c\_id  $\rightarrow$  agent\_no

candidate key = {{c\_id, agent\_no}, {c\_id, br\_code}}

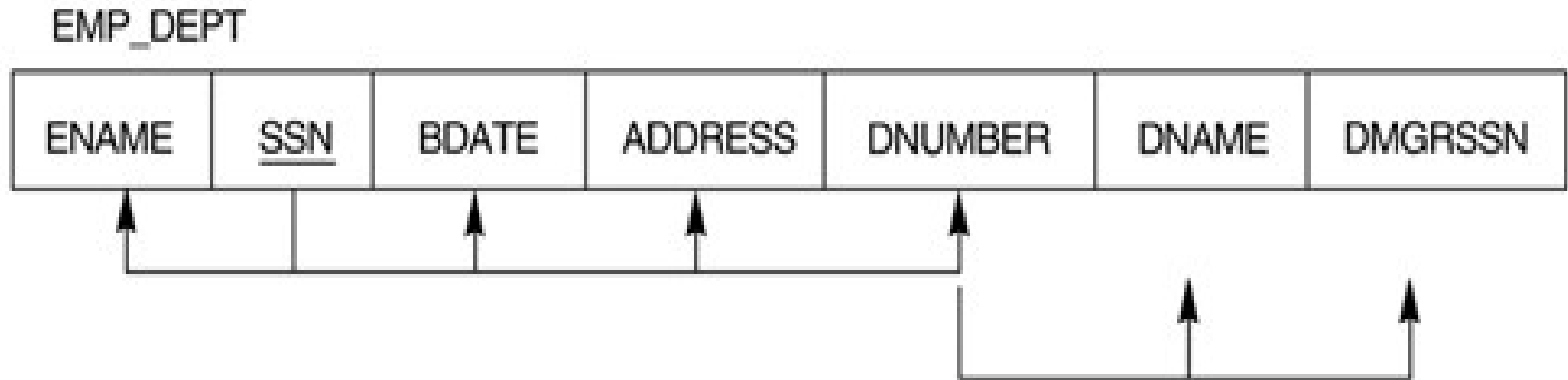
Non prime attribute - **No**

Relation is in 2NF - **Yes**

Redundancy ? - **Yes**

# Third Normal Form

- **Transitive functional dependency:** A functional dependency  $X \rightarrow Z$  that can be derived from two FDs  $X \rightarrow Y$  and  $Y \rightarrow Z$



$SSN \rightarrow DNUMBER, DNUMBER \rightarrow DMGRSSN \quad \square$

$SSN \rightarrow DMGRSSN$  (transitive FD)

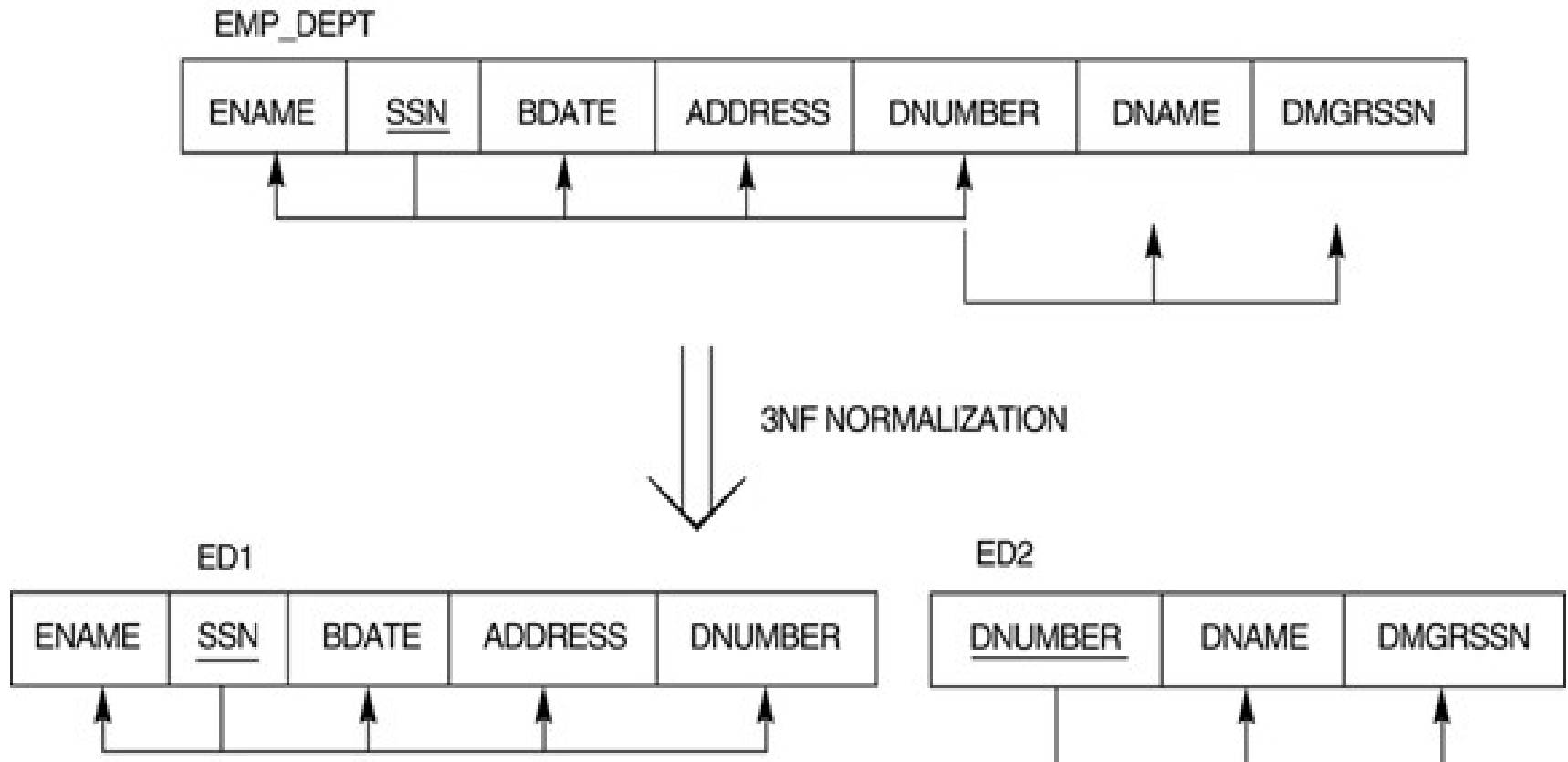
# Third Normal Form

- A relation schema  $R$  is in **third normal form (3NF)** if:
  - It is in 2NF
  - No non-prime attribute  $A$  in  $R$  is transitively dependent on the primary key
- In  $X \rightarrow Y$  and  $Y \rightarrow Z$ , with  $X$  as the primary key, we consider this a problem only if  $Y$  is **not** a candidate key

## **3NF decomposition:**

- Remove all non key attribute(s) which are functionally dependent on other non key attribute(s) from the original relation
- Create a new relation that includes the non key attribute(s) that functionally determines(s) other non-key attribute(s)

# Third Normal Form



# Third Normal Form

see this lecture, and how a redundancy is present.

- **banker**(c\_id, agent\_no, br\_code)

Constraints:

- A agent can work for only one branch
- A agent can serve many customers

agent\_no  $\rightarrow$  br\_code

br\_code, c\_id  $\rightarrow$  agent\_no

candidate key = {{c\_id, agent\_no}, {c\_id, br\_code}}

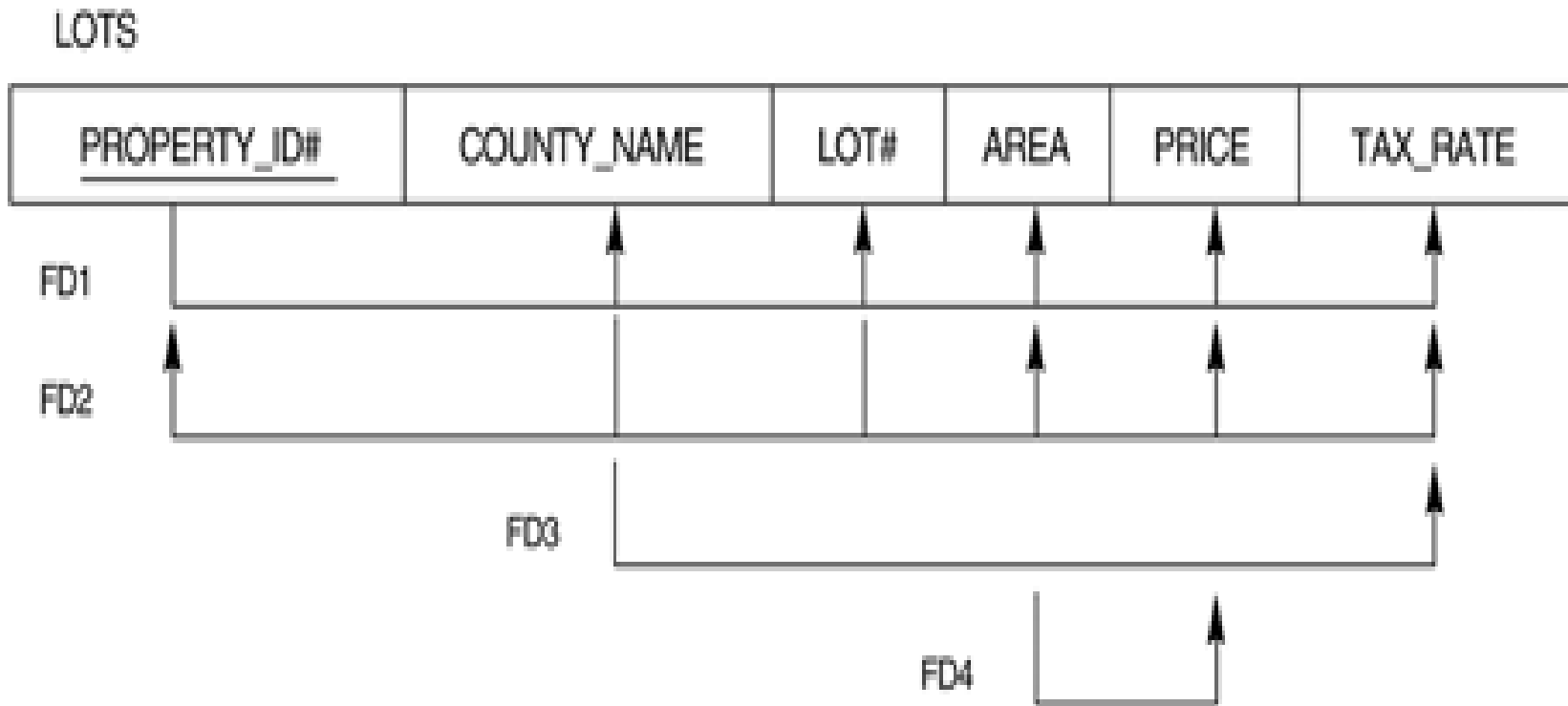
Non prime attribute - **No**

Relation is in 3NF - **Yes**

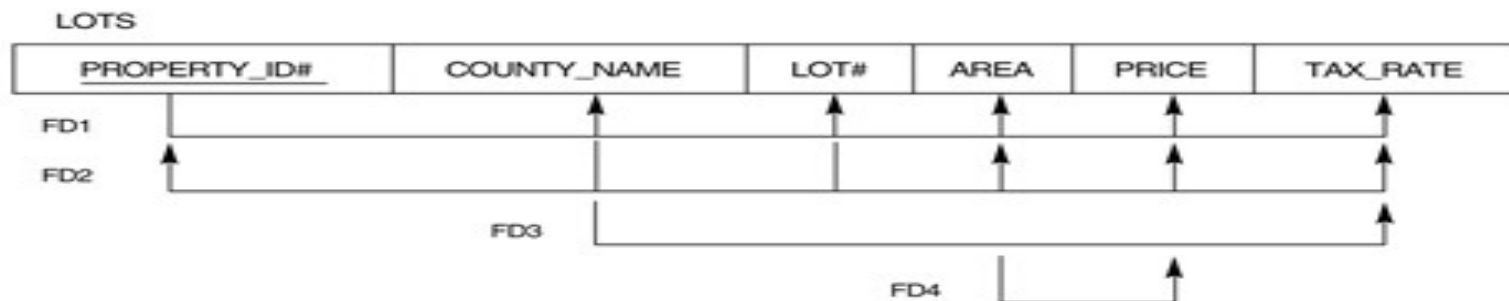
Redundancy ? - **Yes**



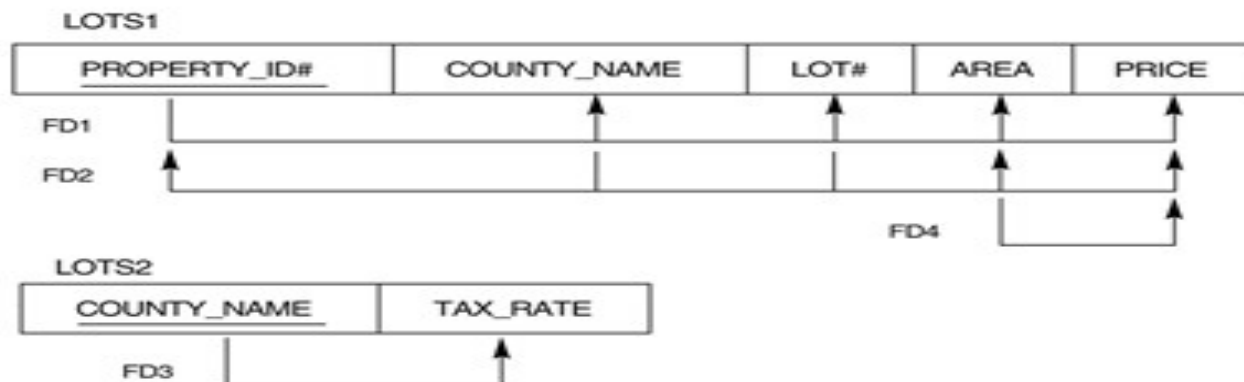
# Practice problem



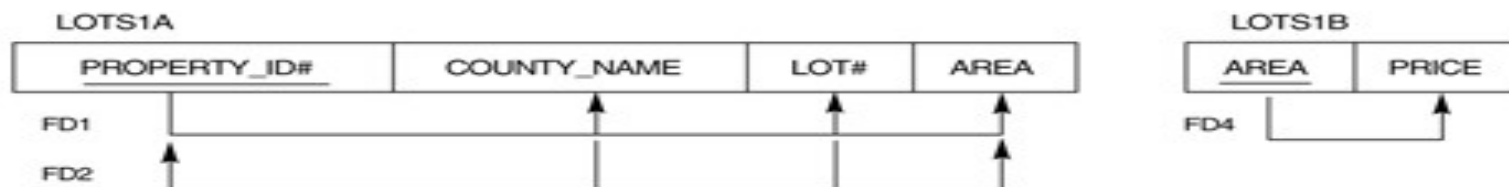
(a)



(b)



(c)



(d)

