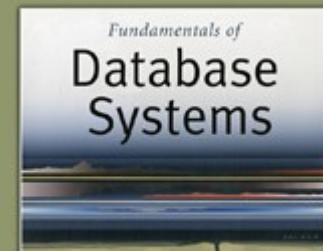


5<sup>th</sup> Edition

Elmasri / Navathe

# Chapter 10

Functional Dependencies and  
Normalization for Relational  
Databases



5<sup>th</sup> Edition

Elmasri / Navathe

# Chapter Outline

- 1 Informal Design Guidelines for Relational Databases
  - 1.1 Semantics 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

- 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?

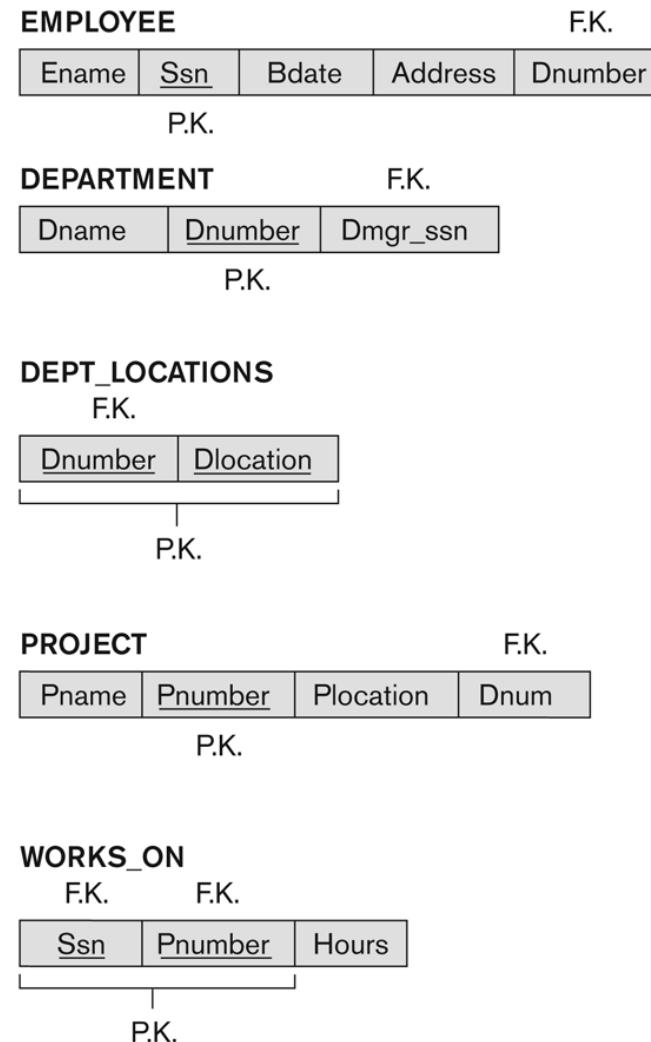
# Informal Design Guidelines for Relational Databases (2)

- 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 10.1**

A simplified COMPANY relational database schema.

# 1.2 Redundant Information in Tuples and Update Anomalies

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

# EXAMPLE OF AN UPDATE ANOMALY

- Consider the relation:
  - $\text{EMP\_PROJ}(\text{Emp\#}, \text{Proj\#}, \text{Ename}, \text{Pname}, \text{No\_hours})$
- 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 INSERT ANOMALY

- Consider the relation:
  - $\text{EMP\_PROJ}(\text{Emp\#}, \text{Proj\#}, \text{Ename}, \text{Pname}, \text{No\_hours})$
- Insert Anomaly:
  - Cannot insert a project unless an employee is assigned to it.
- Conversely
  - Cannot insert an employee unless he/she is assigned to a project.

# EXAMPLE OF AN DELETE ANOMALY

- Consider the relation:
  - $\text{EMP\_PROJ}(\text{Emp\#}, \text{Proj\#}, \text{Ename}, \text{Pname}, \text{No\_hours})$
- 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 10.3**

Two relation schemas suffering from update anomalies.

- (a) EMP\_DEPT and
- (b) EMP\_PROJ.

(a)

**EMP\_DEPT**

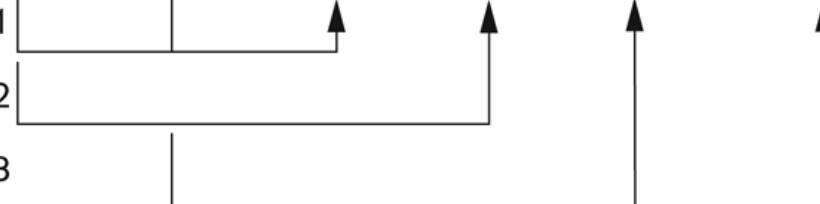
Ename	<u>Ssn</u>	Bdate	Address	Dnumber	Dname	Dmgr_ssn



(b)

**EMP\_PROJ**

<u>Ssn</u>	Pnumber	Hours	Ename	Pname	Plocation
FD1					
FD2					
FD3					



# Figure 10.4 Example States for EMP\_DEPT and EMP\_PROJ

**Figure 10.4**

Example states for EMP\_DEPT and EMP\_PROJ resulting from applying NATURAL JOIN to the relations in Figure 10.2. These may be stored as base relations for performance reasons.

EMP_DEPT							Redundancy	
Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn		
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						Redundancy		Redundancy	
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				

# 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 anomalies 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.

# 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
  - 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

# Functional Dependencies (2)

- $X \rightarrow Y$  holds if whenever two tuples have the same value for  $X$ , they *must have the same value for  $Y$* 
  - For any two tuples  $t_1$  and  $t_2$  in any relation instance  $r(R)$ : If  $t_1[X]=t_2[X]$ , *then*  $t_1[Y]=t_2[Y]$
- $X \rightarrow Y$  in  $R$  specifies a *constraint* on all relation instances  $r(R)$
- Written as  $X \rightarrow Y$ ; 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

# Examples of FD constraints (1)

- Social security number determines employee name
  - $\text{SSN} \rightarrow \text{ENAME}$
- Project number determines project name and location
  - $\text{PNUMBER} \rightarrow \{\text{PNAME}, \text{PLOCATION}\}$
- Employee ssn and project number determines the hours per week that the employee works on the project
  - $\{\text{SSN}, \text{PNUMBER}\} \rightarrow \text{HOURS}$

## 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  $X \rightarrow Y$
  - IR2. (**Augmentation**) If  $X \rightarrow Y$ , then  $XZ \rightarrow YZ$ 
    - (Notation:  $XZ$  stands for  $X \cup Z$ )
  - IR3. (**Transitive**) If  $X \rightarrow Y$  and  $Y \rightarrow Z$ , then  $X \rightarrow Z$
- 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

# Inference Rules for FDs (2)

- Some additional inference rules that are useful:
  - **Decomposition:** If  $X \rightarrowYZ$ , then  $X \rightarrow Y$  and  $X \rightarrow Z$
  - **Union:** If  $X \rightarrow Y$  and  $X \rightarrow Z$ , then  $X \rightarrow YZ$
  - **Pseudotransitivity:** If  $X \rightarrow Y$  and  $WY \rightarrow Z$ , then  $WX \rightarrow Z$
- The last three inference rules, as well as any other inference rules, can be deduced from IR1, IR2, and IR3 (completeness property)

# 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  $F$

## 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 (**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  $F$
- There is an algorithm for checking equivalence of sets of FDs

## 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  $X \rightarrow A$  in F with a dependency  $Y \rightarrow A$ , 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

# 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  $R = \{A_1, A_2, \dots, A_n\}$  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.

## 3.2 First Normal Form

- Disallows
  - composite attributes
  - multivalued attributes
  - **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

(a)

## DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations



(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 10.8**

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.

# Figure 10.9 Normalization nested relations into 1NF

(a)

EMP_PROJ		Projs	
Ssn	Ename	Pnumber	Hours

(b)

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

**Figure 10.9**

Normalizing nested relations into 1NF. (a) Schema of the EMP\_PROJ relation with a *nested relation* attribute PROJS. (b) Example extension of the EMP\_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP\_PROJ into relations EMP\_PROJ1 and EMP\_PROJ2 by propagating the primary key.

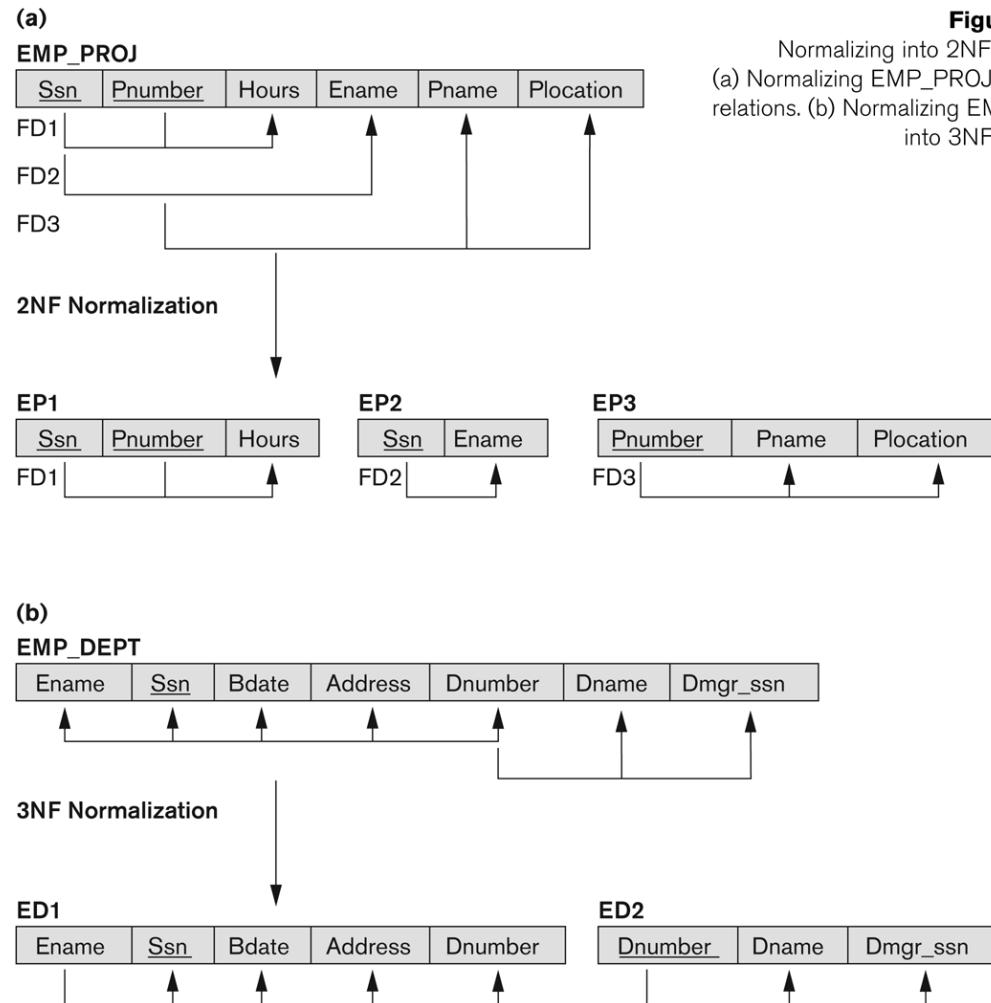
## 3.3 Second Normal Form (1)

- Uses the concepts of **FDs, primary key**
- Definitions
  - **Prime attribute:** An attribute that is member of the primary key K
  - **Full functional dependency:** a FD  $Y \rightarrow Z$  where removal of any attribute from Y means the FD does not hold any more
- Examples:
  - $\{\text{SSN, PNUMBER}\} \rightarrow \text{HOURS}$  is a full FD since neither  $\text{SSN} \rightarrow \text{HOURS}$  nor  $\text{PNUMBER} \rightarrow \text{HOURS}$  hold
  - $\{\text{SSN, PNUMBER}\} \rightarrow \text{ENAME}$  is not a full FD (it is called a partial dependency) since  $\text{SSN} \rightarrow \text{ENAME}$  also holds

# Second Normal Form (2)

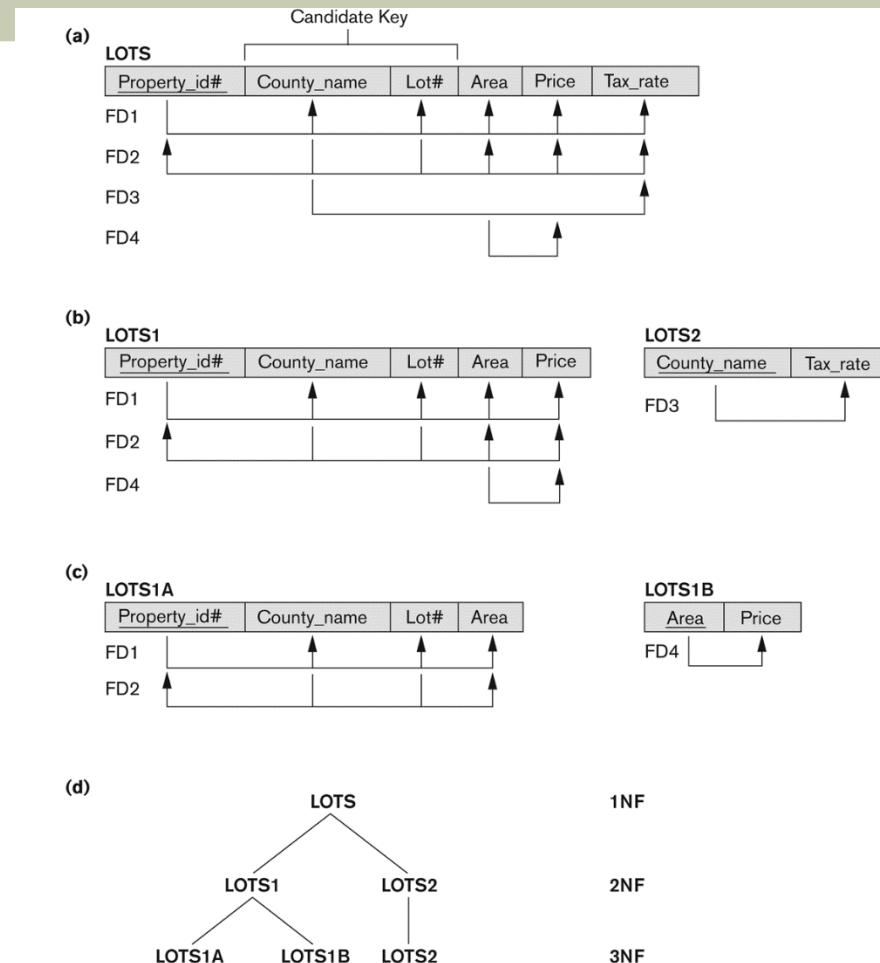
- A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on the primary key
- R can be decomposed into 2NF relations via the process of 2NF normalization

# Figure 10.10 Normalizing into 2NF and 3NF



**Figure 10.10**  
Normalizing into 2NF and 3NF.  
(a) Normalizing EMP\_PROJ into 2NF relations. (b) Normalizing EMP\_DEPT into 3NF relations.

# Figure 10.11 Normalization into 2NF and 3NF



**Figure 10.11**

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

## 3.4 Third Normal Form (1)

- Definition:
  - **Transitive functional dependency:** a FD  $X \rightarrow Z$  that can be derived from two FDs  $X \rightarrow Y$  and  $Y \rightarrow Z$
- Examples:
  - SSN  $\rightarrow$  DMGRSSN is a **transitive FD**
    - Since SSN  $\rightarrow$  DNUMBER and DNUMBER  $\rightarrow$  DMGRSSN hold
  - SSN  $\rightarrow$  ENAME is **non-transitive**
    - Since there is no set of attributes X where SSN  $\rightarrow$  X and X  $\rightarrow$  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  $X \rightarrow Y$  and  $Y \rightarrow Z$ , with X as the primary key, we consider this a problem only if Y is not a candidate key.
  - When Y is a candidate key, there is no problem with the transitive dependency .
  - E.g., Consider EMP (SSN, Emp#, Salary ).
    - Here,  $SSN \rightarrow Emp\# \rightarrow Salary$  and Emp# is a candidate key.

# Normal Forms Defined Informally

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

## 4 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

# General Normal Form Definitions (2)

- Definition:
  - **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  $X \rightarrow A$  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

# 5 BCNF (Boyce-Codd Normal Form)

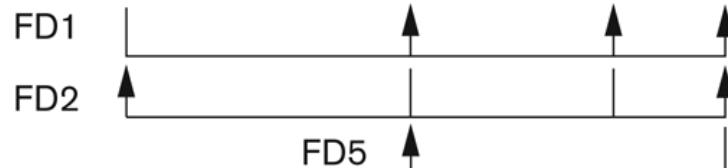
- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever an FD  $X \rightarrow A$  holds in R, then **X is a superkey** of R
- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF
- The goal is to have each relation in BCNF (or 3NF)

# Figure 10.12 Boyce-Codd normal form

(a)

LOTS1A

Property_id#	County_name	Lot#	Area
--------------	-------------	------	------



BCNF Normalization

LOTS1AX

Property_id#	Area	Lot#
--------------	------	------

LOTS1AY

Area	County_name
------	-------------

(b)

R

A	B	C
---	---	---



Figure 10.12

Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.

# Figure 10.13 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
Narayan	Operating Systems	Ammar

**Figure 10.13**

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

# 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
  - {student, instructor} and {student, course}
  - {course, instructor } and {course, student}
  - {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 non-additive (lossless) is discussed in section 11.1.4 under Property LJ1. Verify that the third decomposition above meets the property.

# Chapter Outline

- Informal Design Guidelines for Relational Databases
- Functional Dependencies (FDs)
  - Definition, Inference Rules, Equivalence of Sets of FDs, Minimal Sets of FDs
- Normal Forms Based on Primary Keys
- General Normal Form Definitions (For Multiple Keys)
- BCNF (Boyce-Codd Normal Form)