# 204222 - Fundamentals of Database Systems

## Chapter 15

# Basics of Functional Dependencies and Normalization for Relational Databases

**Adapted for 204222**

**by Areerat Trongratsameethong**

Addison-Wesley
is an imprint of
PEARSON



Copyright © 2011 Pearson Education, Inc. Publishing as Pearson Addison-Wesley

# Chapter 15 Outline

- Informal Design Guidelines for Relation Schemas
- Functional Dependencies
- Normal Forms Based on Primary Keys
- General Definitions of Second and Third Normal Forms
- Boyce-Codd Normal Form

# Introduction

- Levels at which we can discuss *goodness* of relation schemas

  เราสามารถพิจารณาได้ว่า relation schema ดีหรือไม่ดี ได้จาก 2 ระดับ คือ Conceptual Level และ Physical Level

  - Logical (or conceptual) level: ใน Level นี้ ใช้สำหรับสื่อสารกับผู้ใช้ (user) ซึ่งสามารถพิจารณาจาก ความหมายของ attribute หากความหมายของ attribute ชัดเจนจะทำให้ user เข้าใจความหมายของข้อมูลได้ชัดเจนมากขึ้น และจะทำให้ user สามารถบอกข้อมูลที่ต้องการเพื่อการบริหารจัดการข้อมูล หรือข้อมูลที่ต้องการสืบค้นได้ชัดเจน และถูกต้อง
  - Implementation (or physical storage) level
- Approaches to database design: แนวทางในการออกแบบ database ทำได้ 2 ลักษณะ คือ
  - Bottom-up
  - Top-down

# Informal Design Guidelines for Relation Schemas

- Measures of quality
  - Making sure attribute semantics are clear: ให้แน่ใจว่าความหมายของแต่ละ attribute มีความชัดเจน
  - Reducing redundant information in tuples: ลดความซ้ำซ้อนของข้อมูลใน tuples
  - Reducing NULL values in tuples
  - Disallowing possibility of generating spurious (false) tuples: ไม่อนุญาตให้เกิด spurious tuples คือ tuples ที่ผิดพลาด เช่น ได้ข้อมูลเกินจากความเป็นจริงหลังจากรวมข้อมูลจากหลาย Table(ตาราง) หรือ Relation (รีเลชัน)

# Imparting Clear Semantics to Attributes in Relations

- Semantics of a relation
  - Meaning resulting from interpretation of attribute values in a tuple

    ความหมายที่สะท้อนถึง relation มาจากการแปลความหมายของค่าข้อมูล attribute ต่างๆที่อยู่ใน tuple
- Easier to explain semantics of relation
  - Indicates better schema design

    จะต้องอธิบายความหมายของ relation แต่ละ relation ได้ง่าย

# Guideline 1

- Design relation schema so that it is easy to explain its meaning

  ต้องออกแบบให้สามารถอธิบายความหมายของ relation แต่ละ relation ได้ง่าย

- Do not combine attributes from multiple entity types and relationship types into a single relation

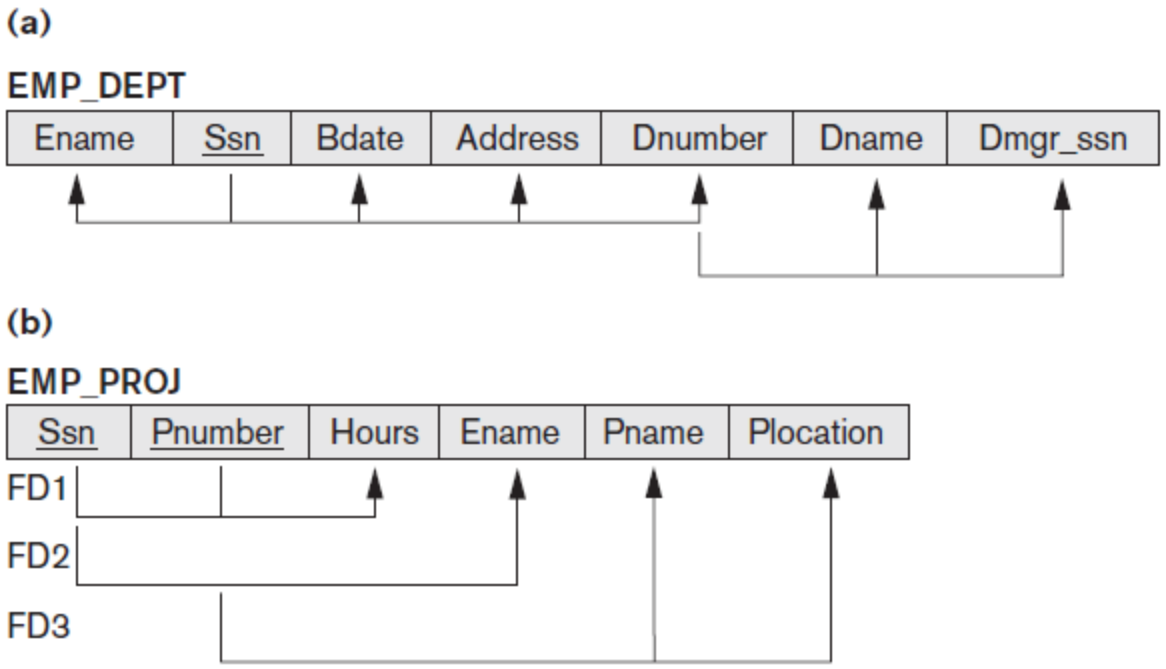
  ต้องไม่รวม attribute ที่มาจากหลาย domain ไว้ใน relation เดียว

- Example of violating Guideline 1: Figure 15.3

  ตัวอย่างการออกแบบที่ผิดพลาด แสดงดังรูปที่ 15.3

# Guideline 1 (cont'd.)

**Figure 15.3**
Two relation schemas suffering from update anomalies. (a) EMP_DEPT and (b) EMP_PROJ.



# Redundant Information in Tuples and Update Anomalies

- Grouping attributes into relation schemas
  - Significant effect on storage space
    การรวมข้อมูลที่มาจากหลาย domain ไว้ด้วยกันมีผลกระทบโดยตรงทำให้เปลืองเนื้อที่ในการจัดเก็บข้อมูล
- Storing natural joins of base relations leads to **update anomalies**
  การจัดเก็บข้อมูลที่อยู่ต่าง domain ถึงแม้จะมีความสัมพันธ์กันในลักษณะ natural joins ก็สามารถนำไปสู่ปัญหา update anomalies ได้ (ปัญหาการปรับปรุงข้อมูลที่ทำให้ข้อมูลมีลักษณะผิดปกติ)
- Types of update anomalies:
  - Insertion
  - Deletion
  - Modification



**Example of anomalies:** http://en.wikipedia.org/wiki/Database_normalization.

# Guideline 2

- Design base relation schemas so that no update anomalies are present in the relations

  ต้องออกแบบไม่ให้เกิด update anomalies ในแต่ละ relation
- If any anomalies are present: ถ้าหลีกเลี่ยงไม่ได้
  - Note them clearly: ให้บันทึกไว้ให้ชัดเจน
  - Make sure that the programs that update the database will operate correctly: ให้แน่ใจว่าโปรแกรมจะปรับปรุงข้อมูลในฐานข้อมูลได้ถูกต้อง

# NULL Values in Tuples

- May group many attributes together into a “fat” relation
  - Can end up with many NULLs

    การรวมข้อมูลจากหลาย domain มาไว้ด้วยกันจะส่งผลให้มีข้อมูลที่มีค่าเป็น NULL เกิดขึ้นใน relation
- Problems with NULLs: ปัญหาการจัดเก็บค่า NULL ทำให้เกิด
  - Wasted storage space: เปลืองเนื้อที่ในการจัดเก็บข้อมูล
  - Problems understanding meaning: การเก็บข้อมูลที่เป็นค่า NULL อาจจะส่งผลในแง่ไม่สื่อความหมาย หรือเข้าใจความหมายได้ยาก

# Guideline 3

- Avoid placing attributes in a base relation whose values may frequently be NULL

  หลีกเลี่ยงการจัดเก็บ attribute ที่มีโอกาสจัดเก็บค่า NULL จำนวนมากๆ
- If NULLs are unavoidable: แต่ถ้าหลีกเลี่ยงไม่ได้
  - Make sure that they apply in exceptional cases only, not to a majority of tuples

    ขอให้เป็นกรณียกเว้น อย่าให้เป็นกรณีส่วนใหญ่ หรือกรณีทั่วไป

# Generation of Spurious Tuples

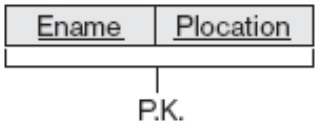
- Figure 15.5(a)
  - Relation schemas EMP_LOCS and EMP_PROJ1

  รูปที่ 15.5 แสดงตัวอย่างการเกิด Spurious Tuples
- NATURAL JOIN
  - Result produces many more tuples than the original set of tuples in EMP_PROJ
  - Called **spurious tuples**

  การรวมข้อมูล (relation) ที่มีความสัมพันธ์กัน แต่เลือก attribute ที่เป็นตัวเชื่อมความสัมพันธ์ไม่เหมาะสม จะทำให้เกิด spurious tuples
  - Represent spurious information that is not valid: การที่มีข้อมูล spurious เกิดขึ้น นั่นหมายความว่า ออกแบบไม่ถูกต้อง

(a)

EMP_LOCS




EMP_PROJ1

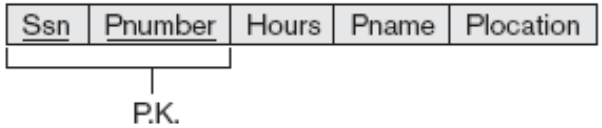



**Figure 15.5**
Particularly poor design for the EMP_PROJ relation in Figure 15.3(b). (a) The two relation schemas EMP_LOCS and EMP_PROJ1. (b) The result of projecting the extension of EMP_PROJ from Figure 15.4 onto the relations EMP_LOCS and EMP_PROJ1.

Update Plocation จาก “Stafford” เป็น “Standford” ต้อง Update กี่ Record?
Update Plocation จาก “Bellaire” เป็น “Houston” ทุก Record เกิดอะไรขึ้น? หากที่จริงต้องการ Update แค่ Plocation ของ Smith

(b)

EMP_LOCS

| Ename | Plocation |
|---|---|
| Smith, John B. | Bellaire |
| Smith, John B. | Sugarland |
| Narayan, Ramesh K. | Houston |
| English, Joyce A. | Bellaire |
| English, Joyce A. | Sugarland |
| Wong, Franklin T. | Sugarland |
| Wong, Franklin T. | Houston |
| Wong, Franklin T. | Stafford |
| Zelaya, Alicia J. | Stafford |
| Jabbar, Ahmad V. | Stafford |
| Wallace, Jennifer S. | Stafford |
| Wallace, Jennifer S. | Houston |
| Borg, James E. | Houston |

EMP_PROJ1

| Ssn | Pnumber | Hours | Pname | Plocation |
|---|---|---|---|---|
| 123456789 | 1 | 32.5 | ProductX | Bellaire |
| 123456789 | 2 | 7.5 | ProductY | Sugarland |
| 666884444 | 3 | 40.0 | ProductZ | Houston |
| 453453453 | 1 | 20.0 | ProductX | Bellaire |
| 453453453 | 2 | 20.0 | ProductY | Sugarland |
| 333445555 | 2 | 10.0 | ProductY | Sugarland |
| 333445555 | 3 | 10.0 | ProductZ | Houston |
| 333445555 | 10 | 10.0 | Computerization | Stafford |
| 333445555 | 20 | 10.0 | Reorganization | Houston |
| 999887777 | 30 | 30.0 | Newbenefits | Stafford |
| 999887777 | 10 | 10.0 | Computerization | Stafford |
| 987987987 | 10 | 35.0 | Computerization | Stafford |
| 987987987 | 30 | 5.0 | Newbenefits | Stafford |
| 987654321 | 30 | 20.0 | Newbenefits | Stafford |
| 987654321 | 20 | 15.0 | Reorganization | Houston |
| 888665555 | 20 | NULL | Reorganization | Houston |

| Pnumber | Pname | Plocation |
|---|---|---|
| 1 | ProductX | Bellaire |
| 2 | ProductY | Sugarland |
| … | … | … |

# Guideline 4

- Design relation schemas to be joined with equality conditions on attributes that are appropriately related
  - Guarantees that no spurious tuples are generated

  ต้องออกแบบให้ relation ที่มีการรวมข้อมูลจาก relation อื่น (Join) โดยเลือก attribute ที่เหมาะสม ซึ่งควรอยู่ในลักษณะ equi join จะทำให้ไม่เกิด spurious tuples
- Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations

  ให้หลีกเลี่ยงการนำ attribute ที่ไม่ใช่ foreign key หรือ primary key มาเป็น ตัวเชื่อมโยง ระหว่าง 2 relation

# Summary and Discussion of Design Guidelines

- Anomalies cause redundant work to be done
- Waste of storage space due to NULLs
- Difficulty of performing operations and joins due to NULL values
- Generation of invalid and spurious data during joins

# Functional Dependencies

- Formal tool for analysis of relational schemas
- Enables us to detect and describe some of the above-mentioned problems in precise terms
- Theory of functional dependency: Constraint between two sets of attributes from the database

> **Definition.** A **functional dependency**, denoted by $X \rightarrow Y$, between two sets of attributes $X$ and $Y$ that are subsets of $R$ specifies a *constraint* on the possible tuples that can form a relation state $r$ of $R$. The constraint is that, for any two tuples $t_1$ and $t_2$ in $r$ that have $t_1[X] = t_2[X]$, they must also have $t_1[Y] = t_2[Y]$.

- Property of semantics or meaning of the attributes
- **Legal relation states**
  - Satisfy the functional dependency constraints

  Relation จะอยู่ในสถานะที่ถูกต้อง/ถูกกฎ เมื่อผ่านข้อกำหนด FD

# Definition of Functional Dependency

- Given a populated relation
  - Cannot determine which FDs hold and which do not
  - Unless meaning of and relationships among attributes known

    เราจะไม่มีทางรู้ว่า Relation แต่ละ Relation จะผ่านข้อกำหนดของ FD หรือไม่จนกว่าเราจะรู้ความหมายของ Attribute และความสัมพันธ์ระหว่าง Attribute วิธีที่จะทำให้รู้คือต้องลองยกตัวอย่างข้อมูลของแต่ละ Relation
  - Can state that FD does not hold if there are tuples that show violation of such an FD

    หลังจากที่รู้ข้อมูลแล้ว หากพบแม้แต่ 1 Record/Tuple ที่ละเมิดข้อกำหนดของ FD จะถือว่า ไม่ผ่านข้อกำหนดของ FD ดังกล่าว

# Normal Forms Based on Primary Keys

- Normalization process: Approaches for relational schema design
  - Perform a conceptual schema design using a conceptual model then map conceptual design into a set of relations

    → Mapping
  - Design relations based on external knowledge derived from existing implementation of files or forms or reports

    → หาข้อมูลจากแบบฟอร์ม หรือรายงาน

# Normalization of Relations

- Takes a relation schema through a series of tests
  - Certify whether it satisfies a certain normal form
  - Proceeds in a top-down fashion

  → ทดสอบแต่ละ Relation ว่าอยู่ในรูปแบบที่เป็นบรรทัดฐานหรือไม่ (Normal Form) ทำในลักษณะ บนลงล่าง เริ่มจาก First Normal Form

- **Normal form tests**

> **Definition.** The **normal form** of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized.

- การทดสอบรูปแบบบรรทัดฐานของแต่ละ Relation หมายถึง รูปแบบขั้นสูงสุดที่ Relation นั้นๆ ผ่านเงื่อนไขการทดสอบ

# Normalization of Relations (cont'd.)

- **Properties that the relational schemas should have:** คุณสมบัติที่แต่ละ Relation ควรจะมี
- **Nonadditive join property,** which guarantees that the spurious tuple generation problem does not occur with respect to the relation schemas created after decomposition.
  → คือ ต้องรับประกันว่าจะไม่เกิด spurious tuple (tuple ที่ผิดพลาด หรือแปลกปลอม) หลังจากที่ decompose (แยก Relation จากขบวนการ Normalization)
  - Extremely critical → คุณสมบัตินี้มีความสำคัญเป็นอย่างยิ่ง
- **Dependency preservation property,** which ensures that each functional dependency is represented in some individual relation resulting after decomposition
  → คือ ต้องมั่นใจว่า แต่ละ FD ก่อนและหลังจากแยก Relation ที่เกิดจากขบวนการ Normalization ยังคงอยู่
  - Desirable but sometimes sacrificed (ยอมสละ) for other factors

# Practical Use of Normal Forms

- Normalization carried out in practice
  - Resulting designs are of high quality and meet the desirable properties stated previously

  → ขบวนการ Normalization ส่งผลให้การออกแบบมีคุณภาพสูง และได้รับทั้ง 2 คุณสมบัติที่กล่าวมาก่อนหน้า
  - Pays particular attention to normalization only up to 3NF, BCNF, or at most 4NF

  → การทำขบวนการ Normalization โดยปกติจะทำถึงขั้นที่ 3 (3NF) BCNF หรืออย่างมากสุดก็ 4NF
- Do not need to normalize to the highest possible normal form → บางครั้งก็ไม่จำเป็นต้องทำขบวนการ Normalization ถึงขั้นสูงสุดที่สามารถทำได้ ในกรณีที่ต้องการความเร็วในการประมวลผลข้อมูล ก็จำเป็นต้องทำ Denormalization คือยอมที่จะเก็บข้อมูลซ้ำซ้อนที่เกิดจากการรวม (Join) ข้อมูลจากหลายตาราง และสร้างเป็นตารางใหม่ ที่มีรูปแบบของบรรทัดฐานขั้นต่ำกว่า

**Definition. Denormalization** is the process of storing the join of higher normal form relations as a base relation, which is in a lower normal form.

# Definitions of Keys and Attributes Participating in Keys

- Definition of **superkey** and **key**
- **Candidate key**
  - If more than one key in a relation schema
    - One is **primary key**
    - Others are **secondary keys**

**Definition.** An attribute of relation schema $R$ is called a **prime attribute** of $R$ if it is a member of *some candidate key* of $R$. An attribute is called **nonprime** if it is not a prime attribute—that is, if it is not a member of any candidate key.

  - **Prime Attribute** คือ **Attribute** ที่เป็นสมาชิกของ **Candidate Key**
  - **Nonprime Attribute** คือ **Attribute** ที่ไม่ใช่สมาชิกของ **Candidate Key**

# First Normal Form

- Part of the formal definition of a relation in the basic (flat) relational model
- Only attribute values permitted are single **atomic (or indivisible) values**

  → **Relation** จะอยู่ในรูป **1NF** ก็ต่อเมื่อทุก **Attribute** ใน **Relation** อยู่ในรูป **Atomic Values (**ค่าข้อมูลไม่สามารถแบ่งย่อยลงได้อีก**)**
- Techniques to achieve first normal form
  - Remove attribute and place in separate relation
  - Expand the key
  - Use several atomic attributes

  → เทคนิคที่จะทำให้อยู่ในรูป 1NF ในกรณีนี้คือ ทำเหมือน Mapping step 6

# First Normal Form (cont'd.)

- Does not allow **nested relations**
  - Each tuple can have a relation within it

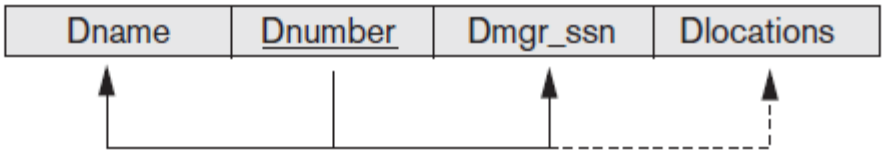
  → อีก 1 ลักษณะคือ ไม่อนุญาตให้มี Nested Relations (Relation ที่มีอีก Relation ซ้อนอยู่ใน Attribute บางตัว)
- To change to 1NF: การเปลี่ยนให้อยู่ในรูป 1NF
  - Remove nested relation attributes into a new relation
  - Propagate the primary key into it

  → ทำเหมือน Mapping Step 6
  - **Unnest** relation into a set of 1NF relations

  → ดู Slide หน้าถัดไป

(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 15.9**
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Sample state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

# Second and Third Normal Form

## Second Normal Form

- Based on concept of **full functional dependency**
  - Versus **partial dependency**

> **Definition.** A relation schema $R$ is in 2NF if every nonprime attribute $A$ in $R$ is *fully functionally dependent* on the primary key of $R$.

- Second normalize into a number of 2NF relations
  - Nonprime attributes are associated only with part of primary key on which they are fully functionally dependent

## Third Normal Form

- Based on concept of transitive dependency

> **Definition.** 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 the primary key.

- Problematic FD
  - Left-hand side is part of primary key
  - Left-hand side is a nonkey attribute

# General Definitions of Second and Third Normal Forms

**Table 15.1** 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). |

# General Definitions of Second and Third Normal Forms (cont'd.)

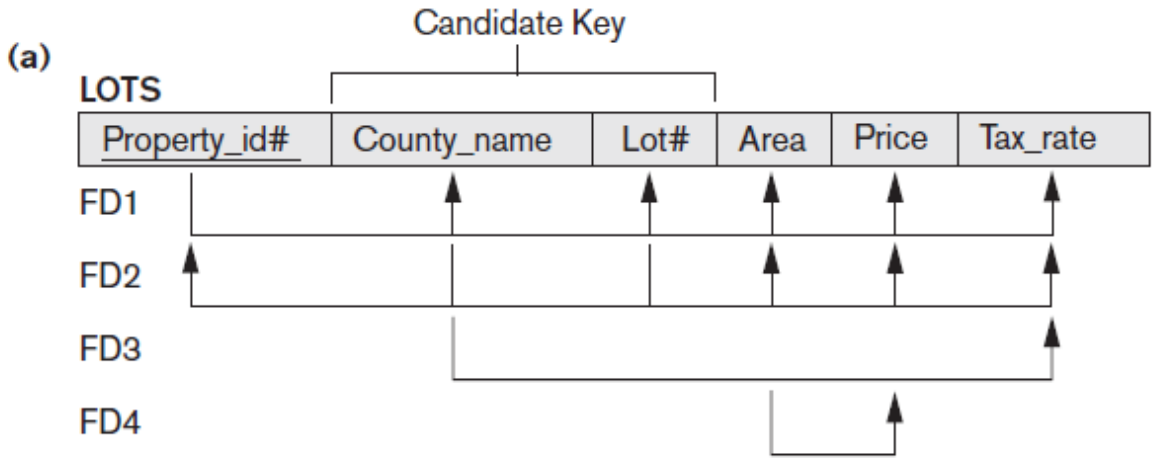
- **Prime attribute**
  - Part of any candidate key will be considered as prime
- Consider partial, full functional, and transitive dependencies with respect to all candidate keys of a relation

# General Definition of Second Normal Form

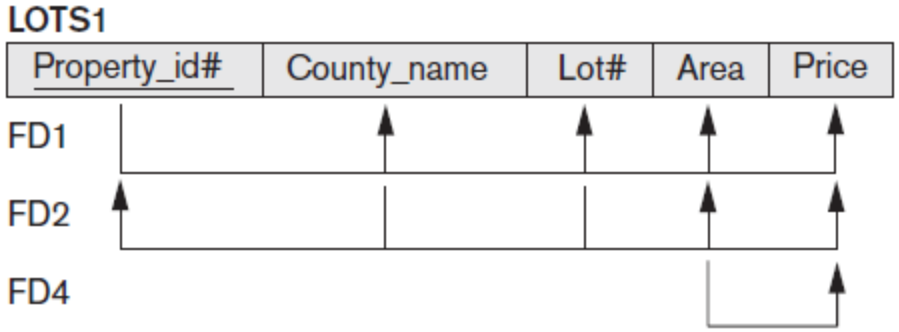
**Definition.** A relation schema $R$ is in **second normal form** (2NF) if every non-prime attribute $A$ in $R$ is not partially dependent on *any* key of $R$.[11]

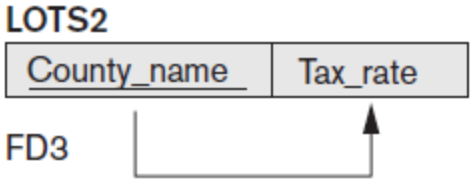
---

**Figure 15.12**
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.

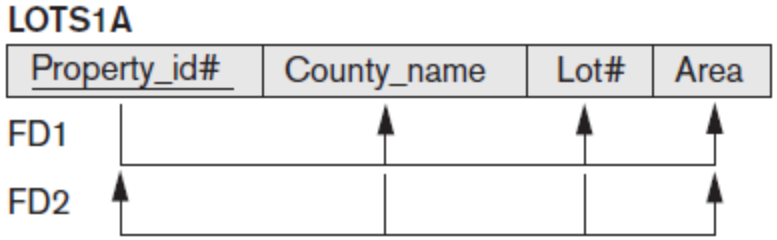


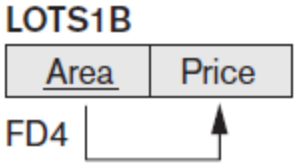
(b)


LOTS1
Property_id#
County_name
Lot#
Area
Price
FD1
FD2
FD4

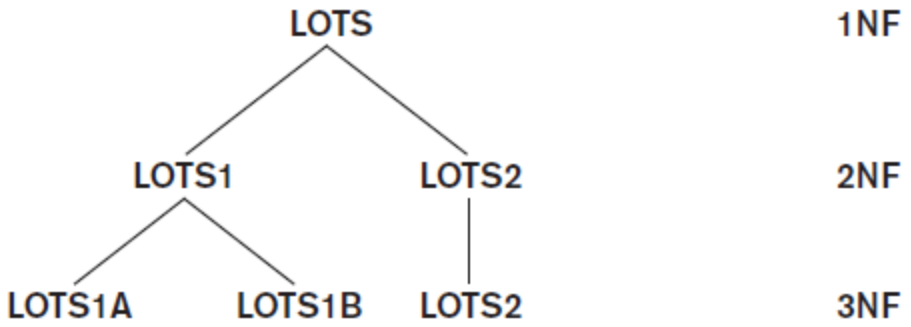

LOTS2
County_name
Tax_rate
FD3

(c)


LOTS1A
Property_id#
County_name
Lot#
Area
FD1
FD2


LOTS1B
Area
Price
FD4

(d)


LOTS
LOTS1
LOTS2
LOTS1A
LOTS1B
LOTS2
1NF
2NF
3NF

# General Definition of Third Normal Form

**Definition.** A relation schema $R$ is in **third normal form** (3NF) if, whenever a *nontrivial* functional dependency $X \rightarrow A$ holds in $R$, either (a) $X$ is a superkey of $R$, or (b) $A$ is a prime attribute of $R$.

**Alternative Definition.** A relation schema $R$ is in 3NF if every nonprime attribute of $R$ meets both of the following conditions:

- It is fully functionally dependent on every key of $R$.
- It is nontransitively dependent on every key of $R$.

# Boyce-Codd Normal Form

- Every relation in BCNF is also in 3NF
  - Relation in 3NF is not necessarily in BCNF

> **Definition.** A relation schema $R$ is in **BCNF** if whenever a *nontrivial* functional dependency $X \rightarrow A$ holds in $R$, then $X$ is a superkey of $R$.

- Difference:
  - Condition which allows A to be prime is absent from BCNF
- Most relation schemas that are in 3NF are also in BCNF

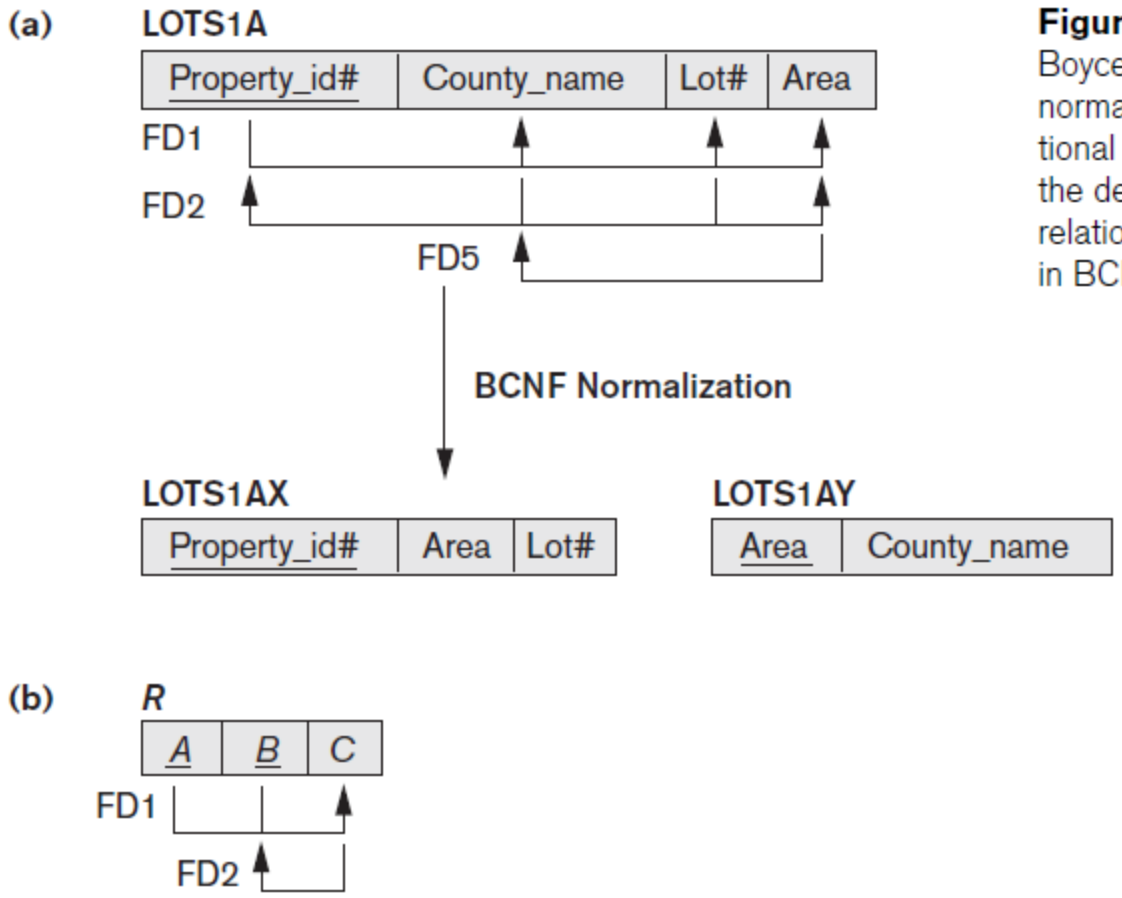



**Figure 15.13**
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.

# Properties of Relational Decompositions (1)

## Testing Binary Decompositions for Lossless Join Property:

- **Binary Decomposition**: decomposition of a relation *R* into two relations.
- **PROPERTY LJ1 (lossless join test for binary decompositions):** A decomposition $D = \{R_1, R_2\}$ of *R* has the lossless join property with respect to a set of functional dependencies *F* on *R* *if and only if* either
  - The FD $((R_1 \cap R_2) \rightarrow (R_1 - R_2))$ is in $F^+$, or
  - The FD $((R_1 \cap R_2) \rightarrow (R_2 - R_1))$ is in $F^+$.

**Note:** $F^+$ is the (complete) set of all dependencies (functional or multivalued) that will hold in every relation state *r* of *R* that satisfies *F*. It is also called the **closure** of *F*.

# Properties of Relational Decompositions (2)

**An example of testing Binary Decompositions for Lossless Join Property**

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 |

Three possible decompositions for relation TEACH

{student, instructor} and {student, course}

{course, instructor } and {course, student}

{instructor, course } and {instructor, student}

# Properties of Relational Decompositions (3)

**An example of testing Binary Decompositions for Lossless Join Property (Cont'd.)**

Decomposition #1

{student, instructor} and {student, course}

| Student | Instructor |
|---|---|
| Narayan | Mark |
| Smith | Navathe |
| Smith | Ammar |
| Smith | Schulman |
| Wallace | Mark |
| Wallace | Ahamad |
| Wong | Omiecinski |
| Zelaya | Navathe |

| Student | Course |
|---|---|
| Narayan | Database |
| Smith | Database |
| Smith | Operating Systems |
| Smith | Theory |
| Wallace | Database |
| Wallace | Operating Systems |
| Wong | Database |
| Zelaya | Database |

Join

| Student | Instructor | Course |
|---|---|---|
| Narayan | Mark | Database |
| Smith | Navathe | Database |
| Smith | Navathe | Operating Systems |
| Smith | Navathe | Theory |
| Smith | Ammar | Database |
| Smith | Ammar | Operating Systems |
| Smith | Ammar | Theory |
| Smith | Schulman | Database |
| Smith | Schulman | Operating Systems |
| Smith | Schulman | Theory |
| Wallace | Mark | Database |
| Wallace | Mark | Operating Systems |
| Wallace | Ahamad | Database |
| Wallace | Ahamad | Operating Systems |
| Wong | Omiecinski | Database |
| Zelaya | Navathe | Database |

student $\not\rightarrow$ instructor, student $\not\rightarrow$ course

Either

- The FD $((R_1 \cap R_2) \rightarrow (R_1 - R_2))$ is in $F^+$, or
- The FD $((R_1 \cap R_2) \rightarrow (R_2 - R_1))$ is in $F^+$.

***Spurious Tuples are generated***

# Properties of Relational Decompositions (4)

**An example of testing Binary Decompositions for Lossless Join Property (Cont'd.)**

Decomposition #2

{course, instructor } and {course, student}

| Course | Instructor |
|---|---|
| Database | Mark |
| Database | Navathe |
| Operating Systems | Ammar |
| Theory | Schulman |
| Operating Systems | Ahamad |
| Database | Omiecinski |

| Course | Student |
|---|---|
| Database | Narayan |
| Database | Smith |
| Operating Systems | Smith |
| Theory | Smith |
| Database | Wallace |
| Operating Systems | Wallace |
| Database | Wong |
| Database | Zelaya |

Join

| Course | Instructor | Student |
|---|---|---|
| Database | Mark | Narayan |
| Database | Mark | Smith |
| Database | Mark | Wallace |
| Database | Mark | Wong |
| Database | Mark | Zelaya |
| Database | Navathe | Narayan |
| Database | Navathe | Smith |
| Database | Navathe | Wallace |
| Database | Navathe | Wong |
| Database | Navathe | Zelaya |
| Database | Omiecinski | Narayan |
| Database | Omiecinski | Smith |
| Database | Omiecinski | Wallace |
| Database | Omiecinski | Wong |
| Database | Omiecinski | Zelaya |
| … | | |

course $\not\rightarrow$ instructor, course $\not\rightarrow$ student

Either

- The FD $((R_1 \cap R_2) \rightarrow (R_1 - R_2))$ is in $F^+$, or
- The FD $((R_1 \cap R_2) \rightarrow (R_2 - R_1))$ is in $F^+$.

***Spurious Tuples are generated***

# Properties of Relational Decompositions (5)

**An example of testing Binary Decompositions for Lossless Join Property (Cont'd.)**

Decomposition #3

{instructor, course } and {instructor, student}

| Instructor | Course |
|---|---|
| Mark | Database |
| Navathe | Database |
| Ammar | Operating Systems |
| Schulman | Theory |
| Ahamad | Operating Systems |
| Omiecinski | Database |

| Instructor | Student |
|---|---|
| Mark | Narayan |
| Navathe | Smith |
| Ammar | Smith |
| Schulman | Smith |
| Mark | Wallace |
| Ahamad | Wallace |
| Omiecinski | Wong |
| Navathe | Zelaya |

Join

| Instructor | Course | Student |
|---|---|---|
| Mark | Database | Narayan |
| Mark | Database | Wallace |
| Navathe | Database | Smith |
| Navathe | Database | Zelaya |
| Ammar | Operating Systems | Smith |
| Schulman | Theory | Smith |
| Ahamad | Operating Systems | Wallace |
| Omiecinski | Database | Wong |

instructor → course, instructor ↛ student

***No Spurious Tuples***

Either

- The FD $((R_1 \cap R_2) \rightarrow (R_1 - R_2))$ is in $F^+$, or
- The FD $((R_1 \cap R_2) \rightarrow (R_2 - R_1))$ is in $F^+$.

# Properties of Relational Decompositions (6)

- **Lossless (Non-additive) Join Property of a Decomposition:**
  - Definition: Lossless join property: a decomposition D = {R1, R2, ..., Rm} of R has the **lossless (non-additive) join property** with respect to the set of dependencies F on R if, for *every* relation state r of R that satisfies F, the following holds, where * is the natural join of all the relations in D:

$$* (\pi_{R1}(r), ..., \pi_{Rm}(r)) = r$$

  - Note: The word loss in lossless refers to loss of information, not to loss of tuples. In fact, for "loss of information" a better term is "**addition of spurious information**"

# Summary

- Informal guidelines for good design
- Functional dependency
  - Basic tool for analyzing relational schemas
- Normalization:
  - 1NF, 2NF, 3NF, BCNF
  - Lossless Join