Chapter 14: Functional Dependencies and Normalization

CS-6360 Database Design

Chris Irwin Davis, Ph.D.

Email: cid021000@utdallas.edu
Phone: (972) 883-3574
Office: ECSS 4.705

Chapter 15 Outline

1) **Informal Design Guidelines for Relation Schemas**
2) **Functional Dependencies (FD)**
3) **Normal Forms**
    - Normal Forms Based on Primary Keys
    - General Definitions of Second and Third Normal Forms
    - Boyce-Codd Normal Form
    - Multivalued Dependency and Fourth Normal Form
    - Join Dependencies and Fifth Normal Form

Introduction

- In chapters 3-6, various aspects of the **relational model** were presented and the **languages** associated with it.
- Each relational **database schema** consists of a number of **relation schemas**, and each **relation schema** consists of a number of **attributes**.
- So far, we have assumed that attributes are grouped to form a relation schema by using the *common sense* of the database designer or by mapping a database schema design from a conceptual data model such as the ER or Enhanced-ER (EER) data model.

Introduction

- These models make the designer identify **entity types** and **relationship types** and their respective **attributes**, which leads to a natural and logical grouping of the attributes into relations when the mapping procedures discussed in Chapter 9 are followed.
- *<u>However</u>*, we still need a formal way of analyzing why one grouping of attributes into a relation schema may be better than another.
- While discussing database design in Chapters 7-9, we did not develop any measure of "*appropriateness*" or "*goodness*" to measure the quality of the design, other than the intuition of the designer.

Introduction

- Here, we discuss some of the theory that has been developed with the goal of:
  - *Evaluating relational schemas for design quality*
  - That is, to measure formally why one set of groupings of attributes into relation schemas is better than another.

Introduction

- There are **two levels** at which we can discuss *goodness* of relation schemas
  - Logical (or conceptual) level
  - Implementation (or physical storage) level
- Approaches to database design:
  - Bottom-up
  - Top-down

Logical (or Conceptual) Level

- How users interpret the **relation schemas** and the meaning of their attributes.
- Having good relation schemas at this level enables users to understand clearly the meaning of the **data** in the relations, and hence to formulate their **queries** correctly.

Implementation (or Physical Storage) Level

- How the tuples in a base relation are stored and updated.
- This level applies only to schemas of **base relations** — which will be *physically stored as files* — whereas at the **logical level** we are interested in schemas of both base relations and views (virtual relations).
- The relational database design theory developed in this chapter applies mainly to **base relations**, although some criteria of appropriateness also apply to **views**.

Informal Design Guidelines for Relation Schemas

Informal Design Guidelines for Relation Schemas

- Measures of quality
  1) Making sure attribute semantics are clear
  2) Reducing redundant information in tuples
  3) Reducing NULL values in tuples
  4) Disallowing possibility of generating spurious tuples

A Simplified COMPANY Schema

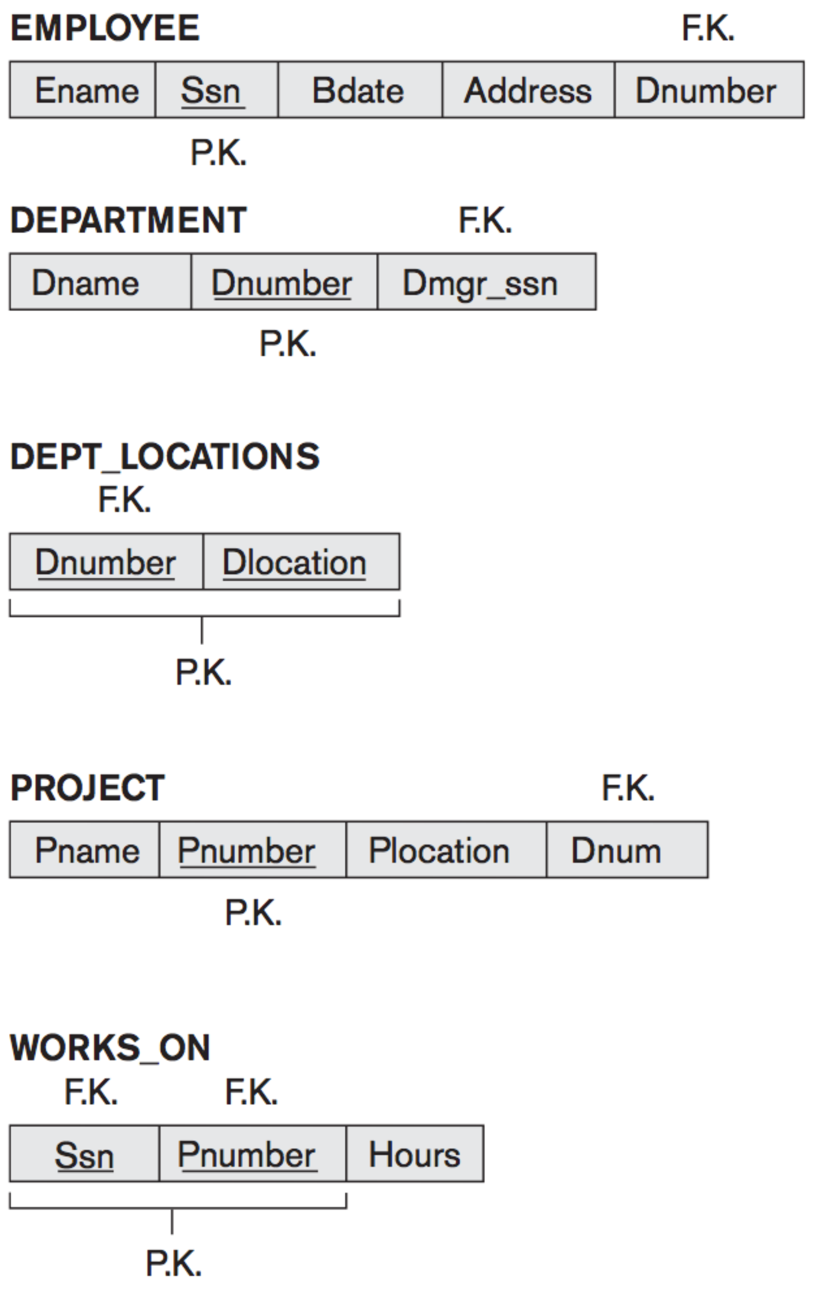


Figure 15.1
A simplified COMPANY relational database schema.

Imparting Clear Semantics to Attributes in Relations

- Semantics of a relation
  - Meaning resulting from interpretation of attribute values in a tuple
- In general, the easier it is to explain semantics of relation, the better the relation schema will be
  - Indicates better schema design

Guideline 1

- Design relation schema so that it is easy to explain its meaning
- Do not combine attributes from multiple entity types and relationship types into a single relation
- Example of violating Guideline 1: Figure 15.3

Guideline 1 (cont'd.)

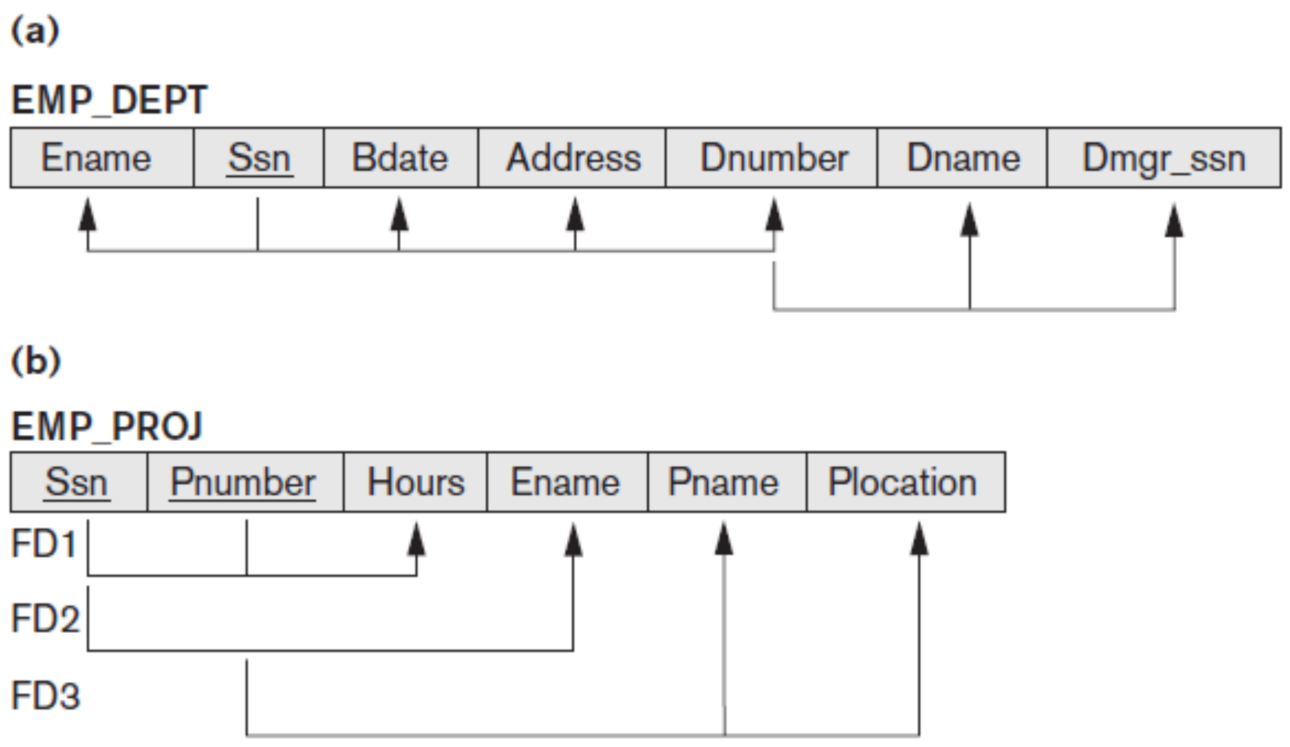


Figure 15.3
Two relation schemas suffering from update anomalies. (a) EMP_DEPT and (b) EMP_PROJ.

Redundant Information in Tuples and Update Anomalies

- Grouping attributes into relation schemas
  - Significant effect on storage space
- Storing natural joins of base relations leads to **update anomalies**
- Types of update anomalies:
  - Insertion
  - Deletion
  - Modification

Insertion Anomalies (two types)

- To insert a new employee tuple into EMP_DEPT, we must include either the attribute values for the department that the employee works for, or NULLs (if the employee does not work for a department as yet).
  - For example, to insert a new tuple for an employee who works in department number 5, we must enter all the attribute values of department 5 correctly so that they are consistent with the corresponding values for department 5 in other tuples in EMP_DEPT.

- In the design of Figure 15.1, we do not have to worry about this consistency problem because we enter only the department number in the employee tuple; all other attribute values of department 5 are recorded only once in the database, as a single tuple in the DEPARTMENT relation.

Insertion Anomalies (two types)

- It is difficult to insert a new department that has no employees as yet in the EMP_DEPT relation.
- The only way to do this is to place NULL values in the attributes for employee. This violates the entity integrity for EMP_DEPT because Ssn is its primary key.
- Moreover, when the first employee is assigned to that department, we do not need this tuple with NULL values any more.
- This problem does not occur in the design of Figure 15.1 because a department is entered in the DEPARTMENT relation whether or not any employees work for it, and whenever an employee is assigned to that department, a corresponding tuple is inserted in EMPLOYEE.

Deletion Anomalies

- The problem of deletion anomalies is related to the second insertion anomaly situation just discussed.
- If we delete from EMP_DEPT an employee tuple that happens to represent the last employee working for a particular department, the information concerning that department is lost from the database.
- This problem does not occur in the database of Figure 15.1 because DEPARTMENT tuples are stored separately.

Modification Anomalies

- In EMP_DEPT, if we change the value of one of the attributes of a particular department—say, the manager of department 5—we must update the tuples of all employees who work in that department; otherwise, the database will become inconsistent.
- If we fail to update some tuples, the same department will be shown to have two different values for manager in different employee tuples, which would be wrong

Update Anomalies

- It is easy to see that these three anomalies are undesirable and cause difficulties to maintain consistency of data as well as require unnecessary updates that can be avoided; hence, we can state the next guideline as follows.

Guideline 2

- Design base relation schemas so that no update anomalies are present in the relations
- If any anomalies *<u>are</u>* present:
  - Identify them
  - Note them clearly
  - Make sure that the programs that update the database will operate correctly

Redundancy

Redundancy

EMP_DEPT

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

Redundancy

Redundancy — Ename

Redundancy — Pname, Plocation

EMP_PROJ

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 3

- Avoid placing attributes in a base relation whose values may frequently be NULL
- If NULLs are unavoidable:
  - Make sure that they apply in exceptional cases only, not to a majority of tuples

NULL Values in Tuples

- In some schema designs we may group many attributes together into a “fat” or “wide” relation
  - Can end up with many NULLs
- Problems with NULLs
  - Wasted storage space
  - Problems understanding meaning
    - Does not apply to this tuple
    - Value for the tuple is unknown
    - Value for the tuple is known, but absent

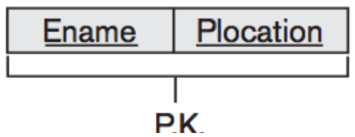
Generation of Spurious Tuples

- Figure 15.5(a)
  - Consider the two relation schemas EMP_LOCS and EMP_PROJ1 in Figure 15.5(a), which can be used instead of the single EMP_PROJ relation in Figure 15.3(b).
- NATURAL JOIN
  - Result produces many more tuples than the original set of tuples in EMP_PROJ
  - Called **spurious tuples**
  - Represent spurious information that is not valid

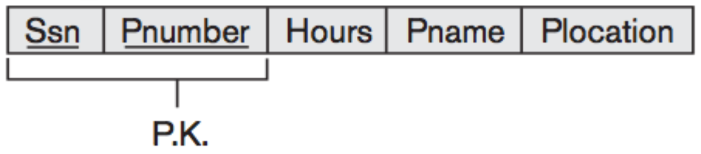
Generation of Spurious Tuples

(a)

EMP_LOCS



EMP_PROJ1



(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

Guideline 4

- Design relation schemas to be joined with **equality conditions** on attributes that are appropriately related
    - Guarantees that no spurious tuples are generated
- Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations

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

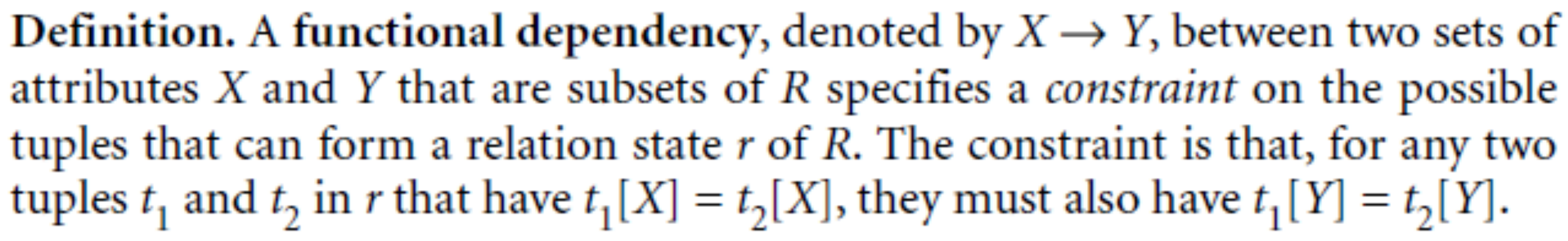
Functional Dependencies

Functional Dependencies

- Formal tool for analysis of relational schemas
  - "*The single most important concept in relational schema design theory is that of a functional dependency.*"
- Enables us to detect and describe some of the above-mentioned problems in precise terms
- Theory of functional dependency

Definition 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

Definition of Functional Dependency

- This means that the values of the *Y* component of a tuple in *r* depend on, or are *determined by*, the values of the *X* component; alternatively, the values of the *X* component of a tuple uniquely (or **functionally**) *determine* the values of the *Y* component.
- We also say that there is a functional dependency from *X* to *Y*, or that *Y* is functionally dependent on *X*. The abbreviation for functional dependency is **FD** or **f.d.** The set of attributes *X* is called the **left-hand side** of the FD, and *Y* is called the **right-hand side**.

Definition of Functional Dependency

- *X* functionally determines *Y* in a relation schema *R* if, and only if, whenever two tuples of *r*(*R*) agree on their *X*-value, they *must* necessarily agree on their *Y*-value.
- Note the following:
  - If a constraint on *R* states that there cannot be more than one tuple with a given *X*-value in any relation instance *r*(*R*) – that is, *X* is a **candidate key** of *R* – this implies that $X \rightarrow Y$ for any subset of attributes *Y* of *R* (because the key constraint implies that no two tuples in any legal state *r*(*R*) will have the same value of *X*). If *X* is a candidate key of *R*, then $X \rightarrow R$.
  - If $X \rightarrow Y$ is in *R*, does not imply whether or not $Y \rightarrow X$ is in *R*, i.e. not bidirectional, not **iff**

Normal Forms Based on Primary Keys

Normal Forms Based on Primary Keys

- Having introduced functional dependencies, we are now ready to use them to specify some aspects of the semantics of relation schemas.
- We assume that a set of functional dependencies is given for each relation, and
  - that each relation has a designated primary key;
  - this information combined with the tests (conditions) for normal forms drives the normalization process for relational schema design.

Normal Forms Based on Primary Keys

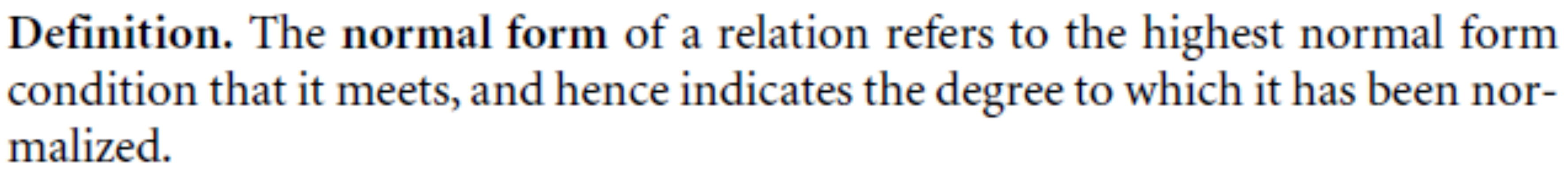
- **Normalization Process** – Most practical relational design projects take one of the following two approaches:
  - Perform a conceptual schema design using a conceptual model then map conceptual design into a set of relations
  - Design relations based on external knowledge derived from existing implementation of files or forms or reports

Normalization of Relations

- The normalization process takes a relation schema through a series of tests to
  - *Certify* whether it satisfies a certain **normal form**
  - Proceeds in a top-down fashion
- The normalization procedure provides database designers with the following:
  - A formal framework for analyzing relation schemas based on their keys and on the functional dependencies among their attributes
  - A series of normal form tests that can be carried out on individual relation schemas so that the relational database can be **normalized** to any desired degree

Normalization of Relations

- **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.

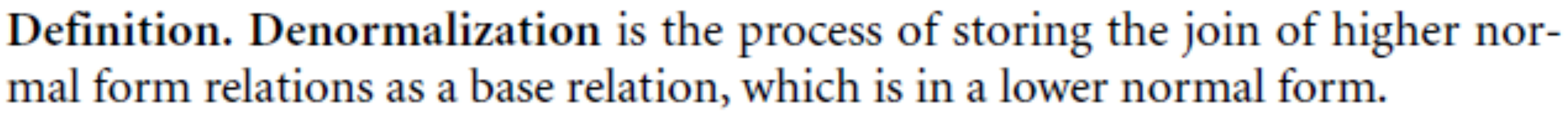
- A relation may not require normalization beyond a certain level to be fully normalized

Normalization of Relations (cont'd.)

- Properties that the relational schemas should have:
  - **Non-additive join property**
    - Extremely critical
    - Guarantees that the spurious tuple generation problem discussed in Section 15.1.4 does not occur with respect to the relation schemas created after decomposition.
  - **Dependency preservation property**
    - Desirable but sometimes sacrificed for other factors
    - Ensures that each functional dependency is represented in some individual relation resulting after decomposition.

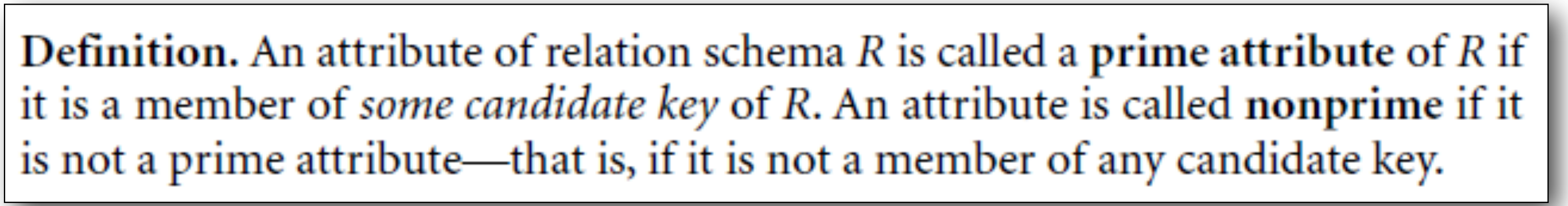
Practical Use of Normal Forms

- Normalization carried out in *current* practice
  - Pays particular attention to normalization only up to 3NF, BCNF, or at most 4NF
  - Resulting designs are of high quality and meet the desirable properties stated previously
- Do not need to normalize to the highest possible normal form
  - Sometimes for performance reasons


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**
  - Removing any attributes results results in *not a key*
  - If more than one key in a relation schema
    - One is **primary key**
    - Others are **secondary keys**
- **Prime attribute**


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.

Normal Forms

First Normal Form

First Normal Form

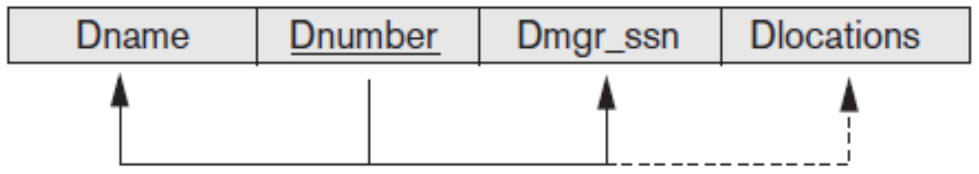
- **1NF** is part of the formal definition of a relation in the basic (flat) relational model
- Only attribute values permitted are single **atomic (or indivisible) values**
- Techniques to achieve first normal form
  - Remove attribute and place in separate relation
  - Expand the key
  - Use several atomic attributes

First Normal Form (cont'd.)

- Does not allow **nested relations**
  - i.e. Each tuple can have a relation within it
- To change to 1NF:
  - Remove nested relation attributes into a new relation
  - Propagate the primary key into it
  - **Unnest** relation into a set of 1NF relations

(a)

DEPARTMENT



(b)

DEPARTMENT

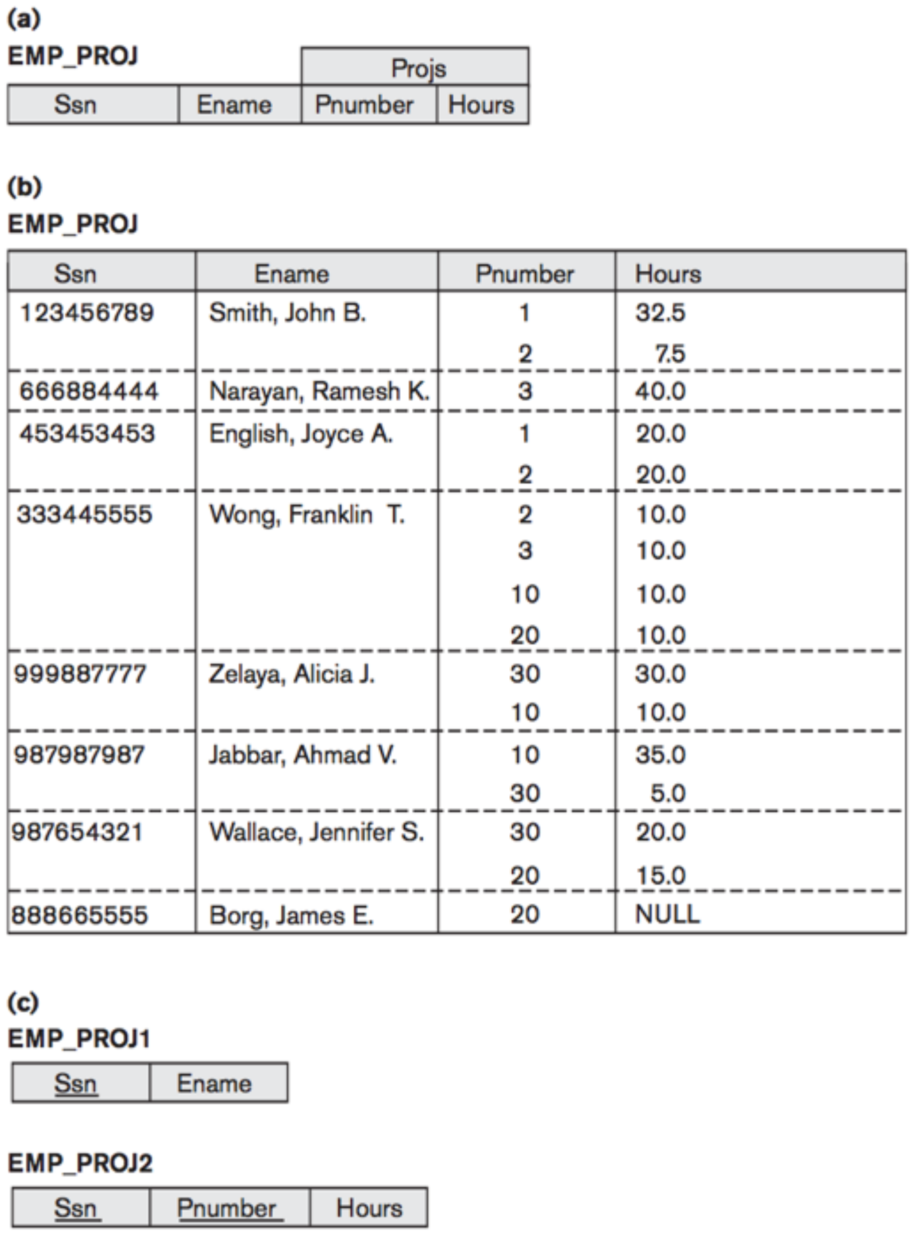
Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(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.



(a)

EMP_PROJ

		Projs	
Ssn	Ename	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

(c)

EMP_PROJ1

<u>Ssn</u>	Ename

EMP_PROJ2

<u>Ssn</u>	<u>Pnumber</u>	Hours

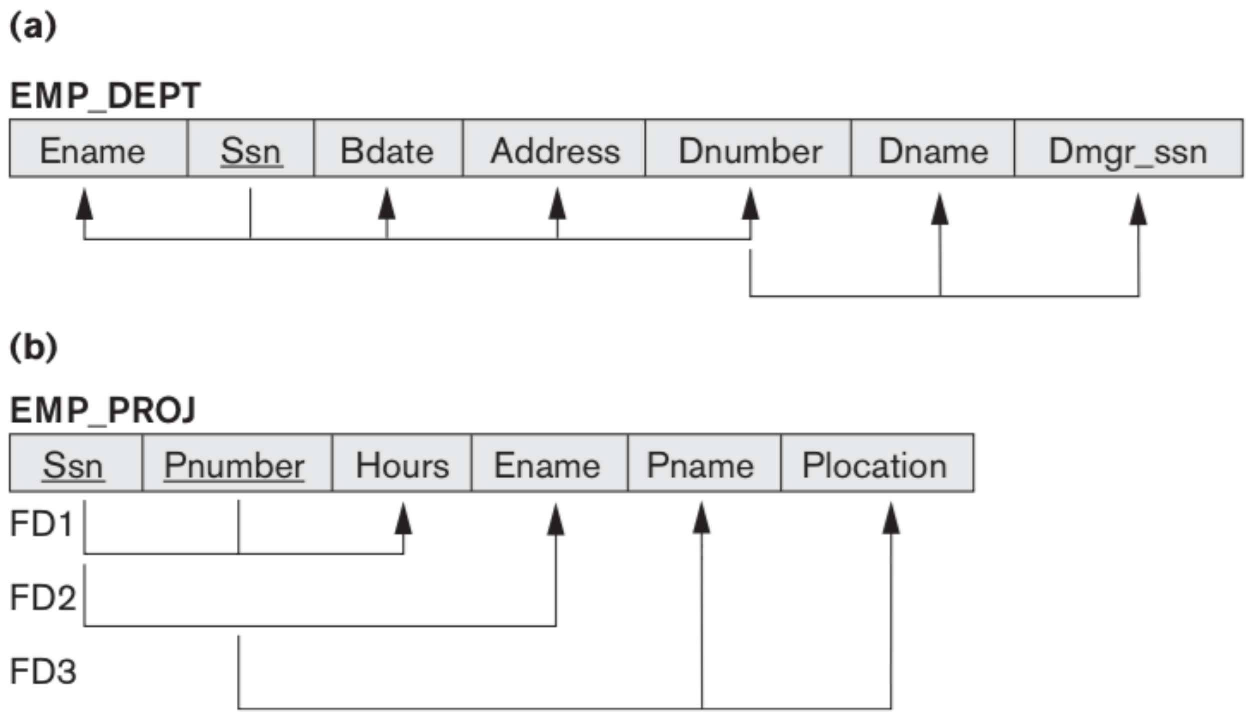
Figure 15.10
Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Sample extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

Second Normal Form

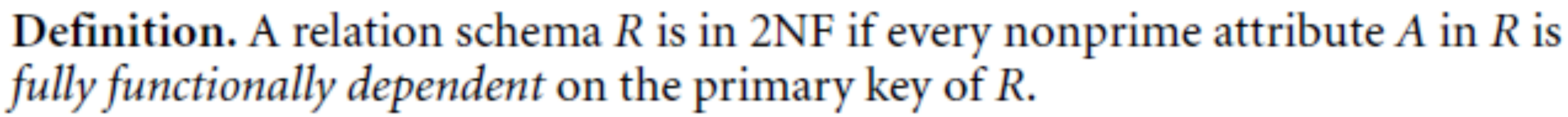
Second Normal Form (2NF)

- **2NF** is based on 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; that is, for any attribute $A \in X$, $(X - \{A\})$ does not functionally determine Y.
- Versus **partial dependency**
  - If some attribute $A \in X$ can be removed from X and the dependency still holds
  - In Figure 15.3(b), {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.

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



Second Normal Form (2NF)



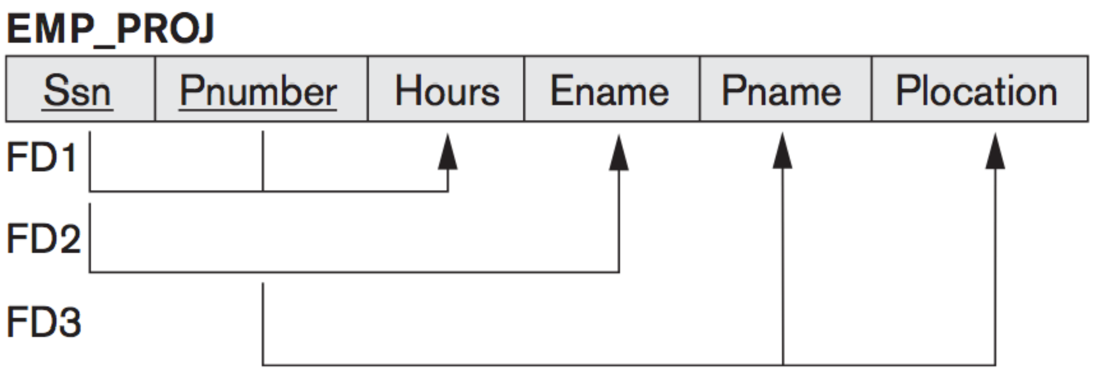
Definition. A relation schema *R* is in 2NF if every nonprime attribute *A* in *R* is *fully functionally dependent* on the primary key of *R*.

- 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

Second Normal Form (2NF)

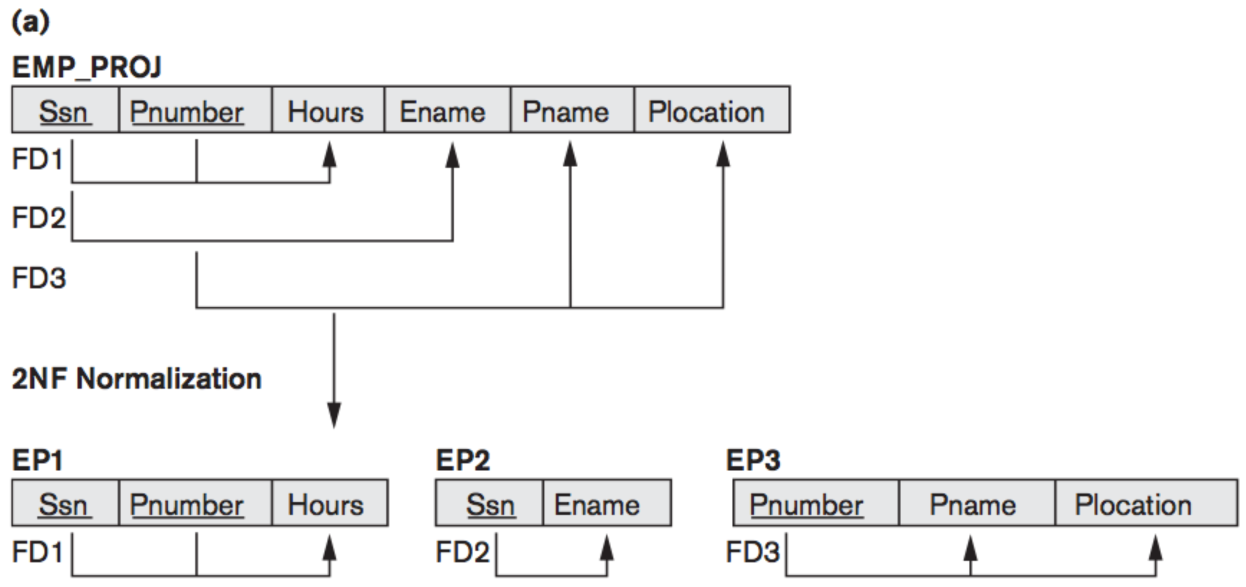
- The EMP_PROJ relation in Figure 15.3(b) is in 1NF but is not in 2NF. The nonprime attribute Ename violates 2NF because of FD2, as do the nonprime attributes Pname and Plocation because of FD3.
- The functional dependencies FD2 and FD3 make Ename, Pname, and Plocation partially dependent on the primary key {Ssn, Pnumber} of EMP_PROJ, thus violating the 2NF test.

(b)



Second Normal Form (2NF)

- 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.
- Therefore, the functional dependencies FD1, FD2, and FD3 in Figure 15.3(b) lead to the decomposition of EMP_PROJ into the three relation schemas EP1, EP2, and EP3 shown in Figure 15.11(a), each of which is in 2NF.

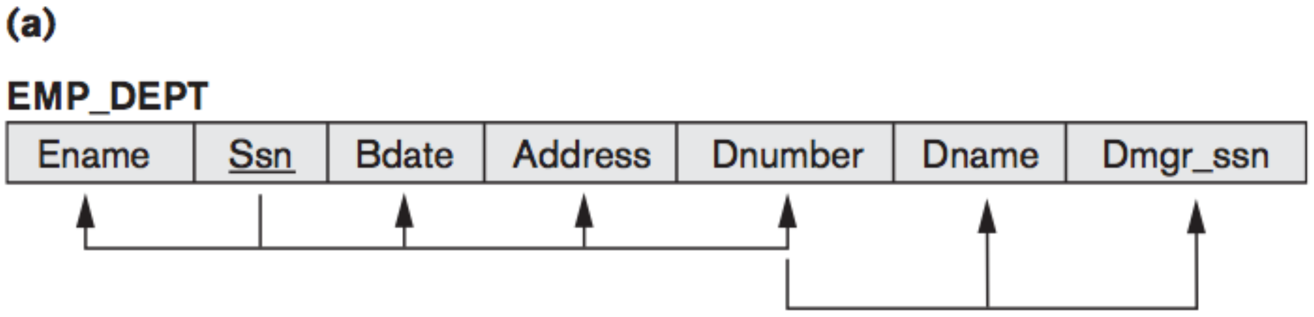


Third Normal Form

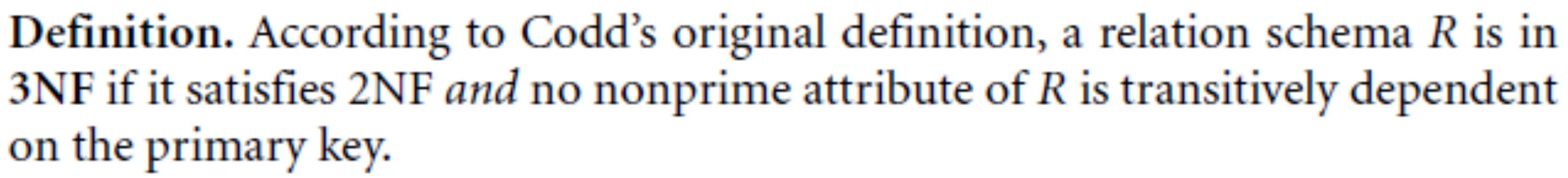
Third Normal Form (3NF)

- **3NF** is based on concept of **transitive dependency**
  - …if there exists a set of attributes *Z* in *R* that is neither a candidate key nor a subset of any key of *R*, and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold.
  - The dependency Ssn → Dmgr_ssn is transitive through Dnumber in EMP_DEPT in Figure 15.3(a), because both the dependencies Ssn → Dnumber and Dnumber → Dmgr_ssn hold and Dnumber is neither a key itself nor a subset of the key of EMP_DEPT.

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



Third Normal Form (3NF)



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 non-key attribute
- The reason why Normalization is a top-down process

Third Normal Form (3NF)

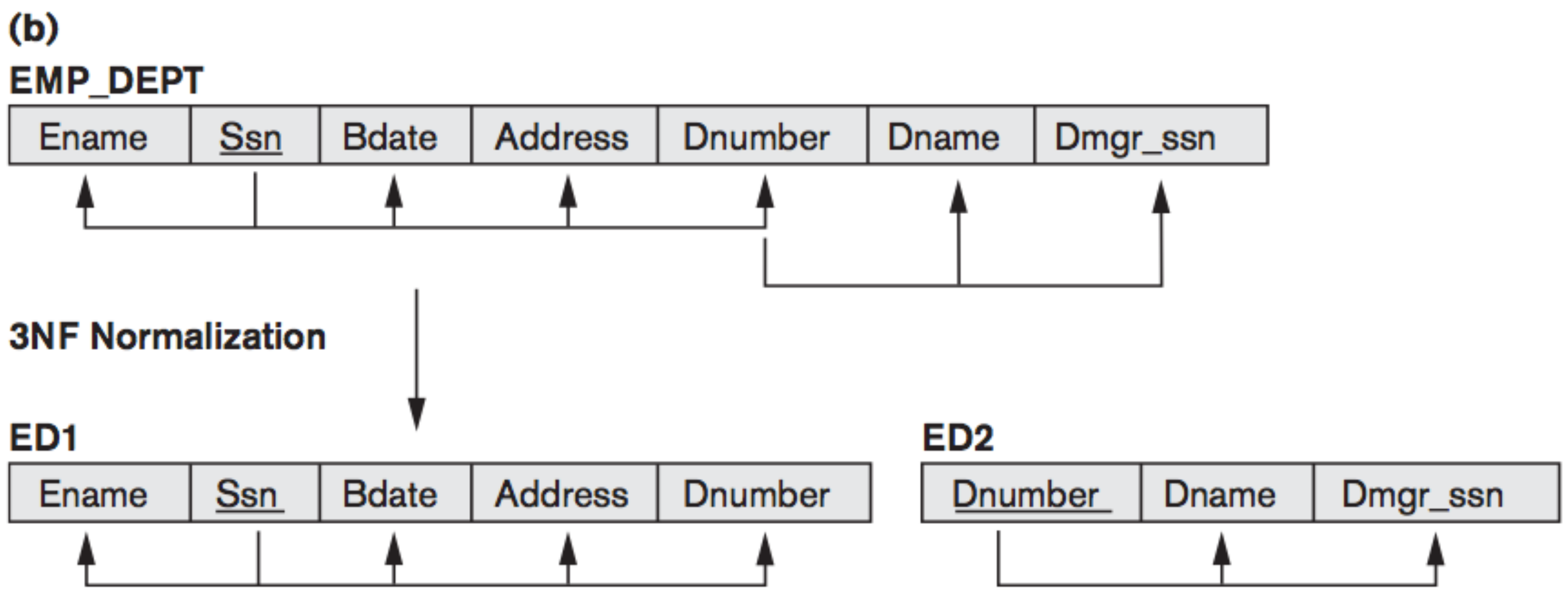


Figure 15.11
Normalizing into 2NF and 3NF. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

General Definitions of Second and Third Normal Forms

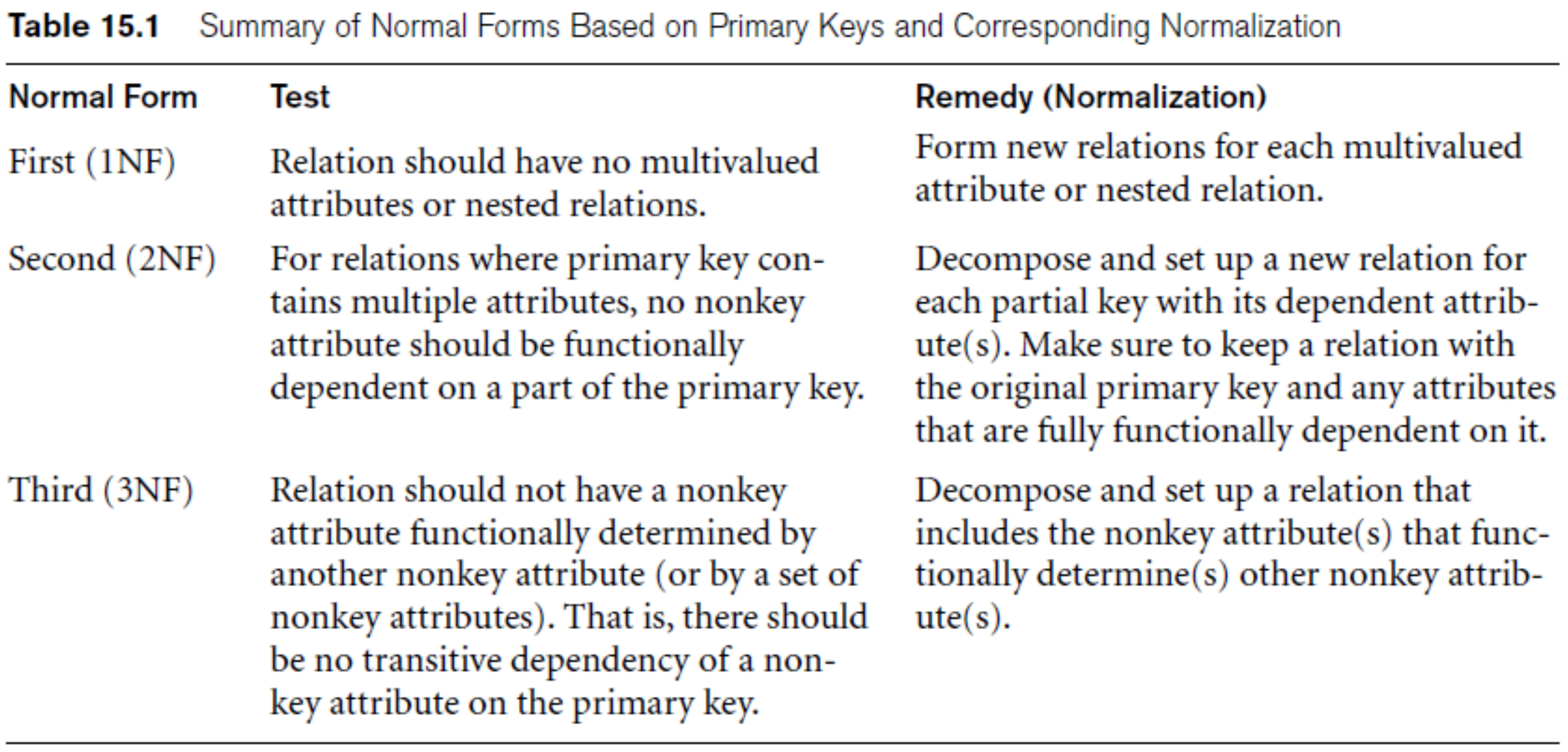


Table 15.1 Summary of Normal Forms Based on Primary Keys and Corresponding Normalization

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 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]

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.

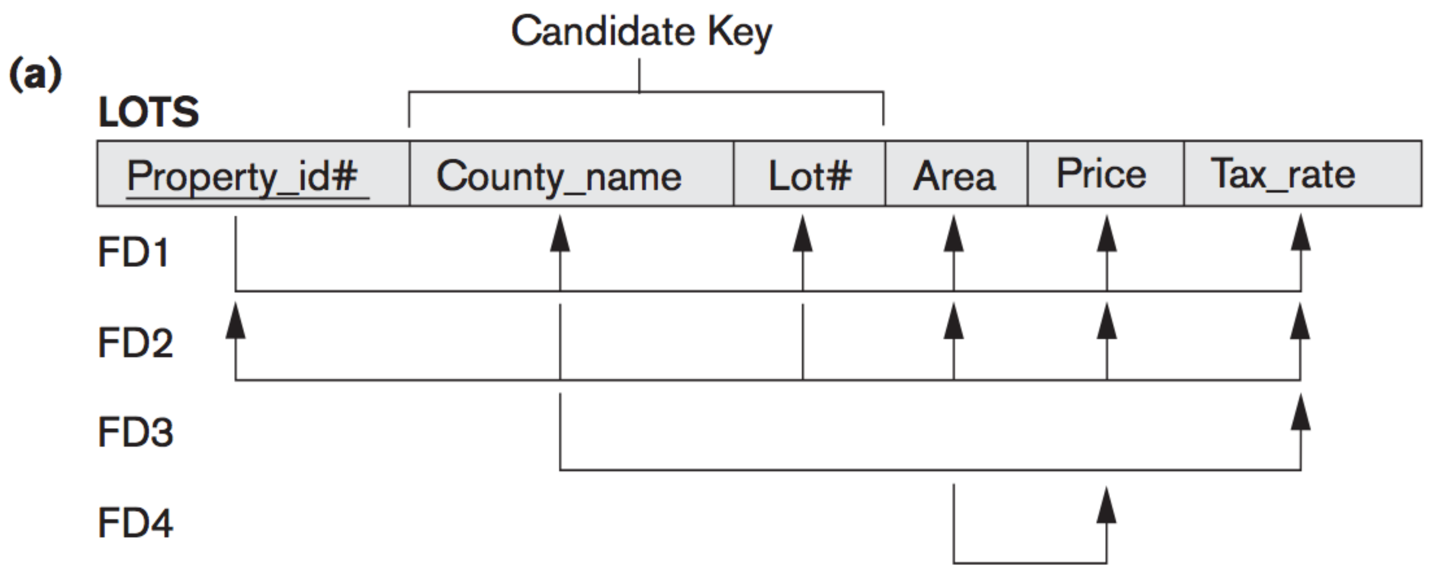
Alternative Definition. A relation schema R is in 3NF if every nonprime attribute of R meets both of the following conditions:

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

Normalization Example

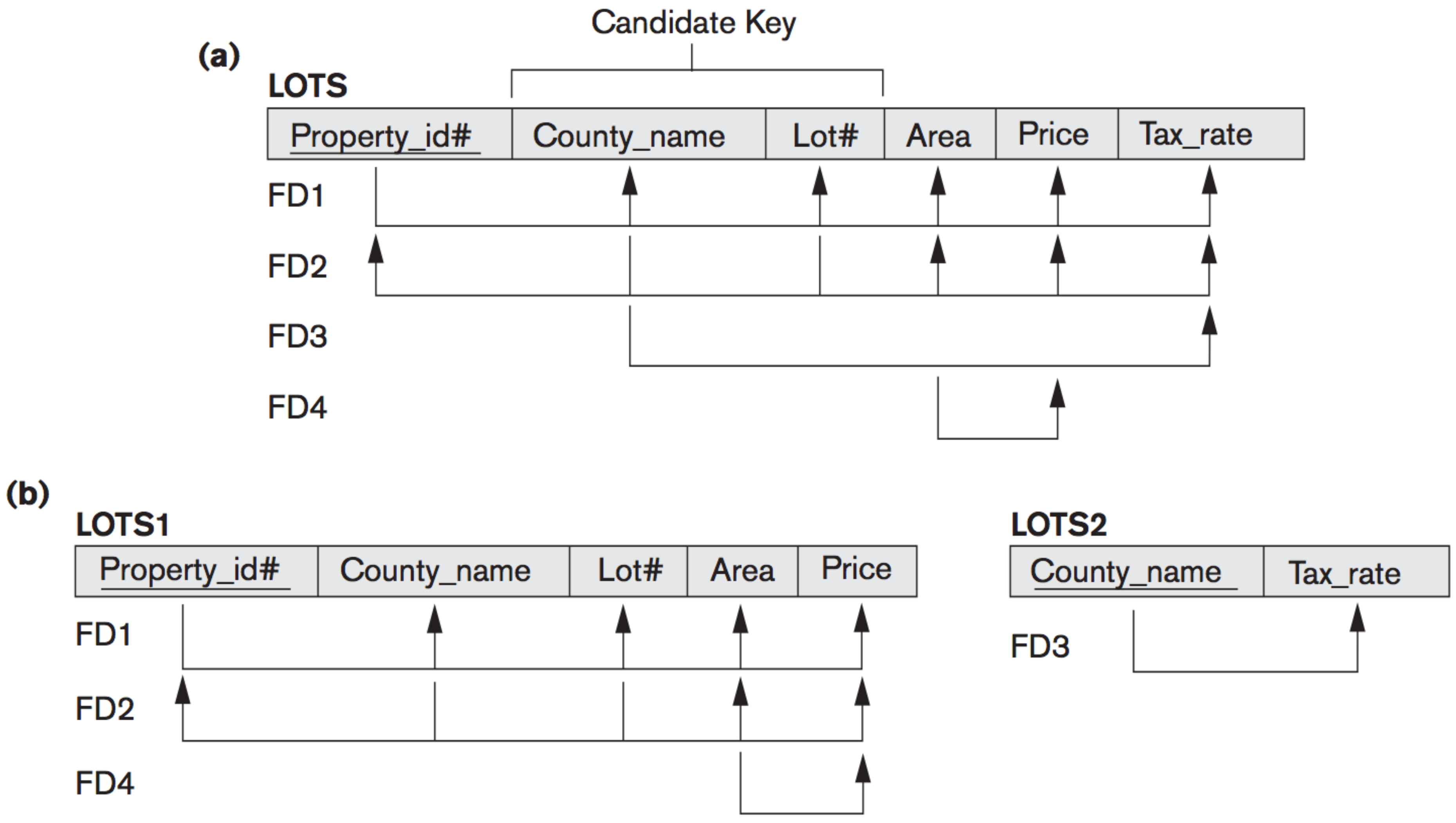
Normalization into 2NF and 3NF

- (a) The LOTS relation with its functional dependencies FD1 through FD4.
- Note: The LOTS relation is already 1NF.



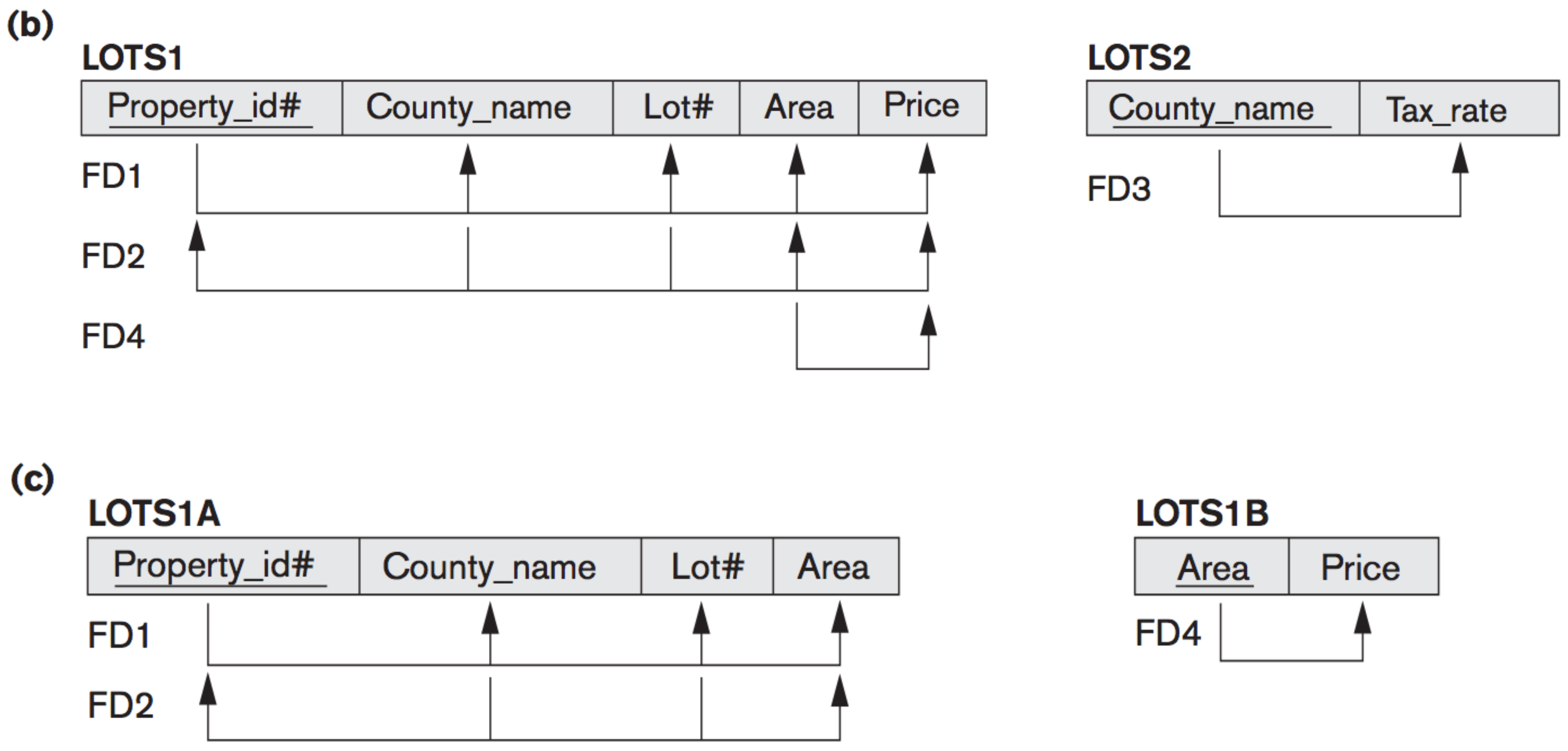
Normalization into 2NF and 3NF

- (b) Decomposing into the 2NF relations LOTS1 and LOTS2.



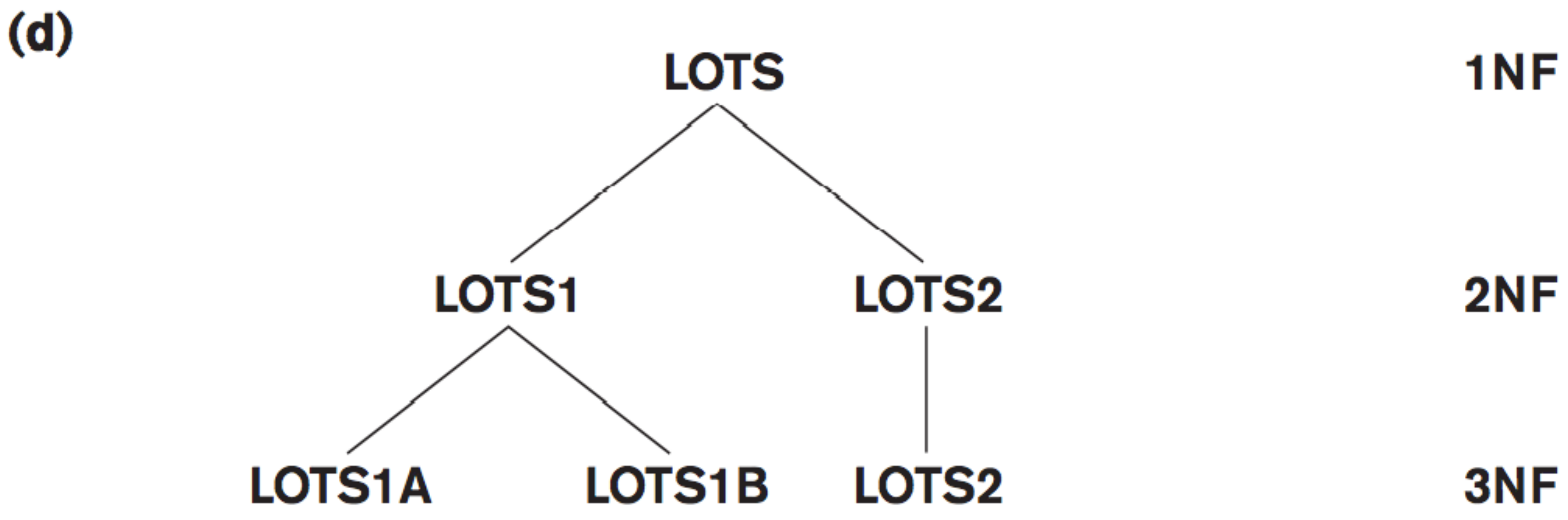
Normalization into 2NF and 3NF

- (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B.



Normalization into 2NF and 3NF

- (d) Summary of the progressive normalization of LOTS.



Boyce-Codd Normal Form

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

Boyce-Codd Normal Form

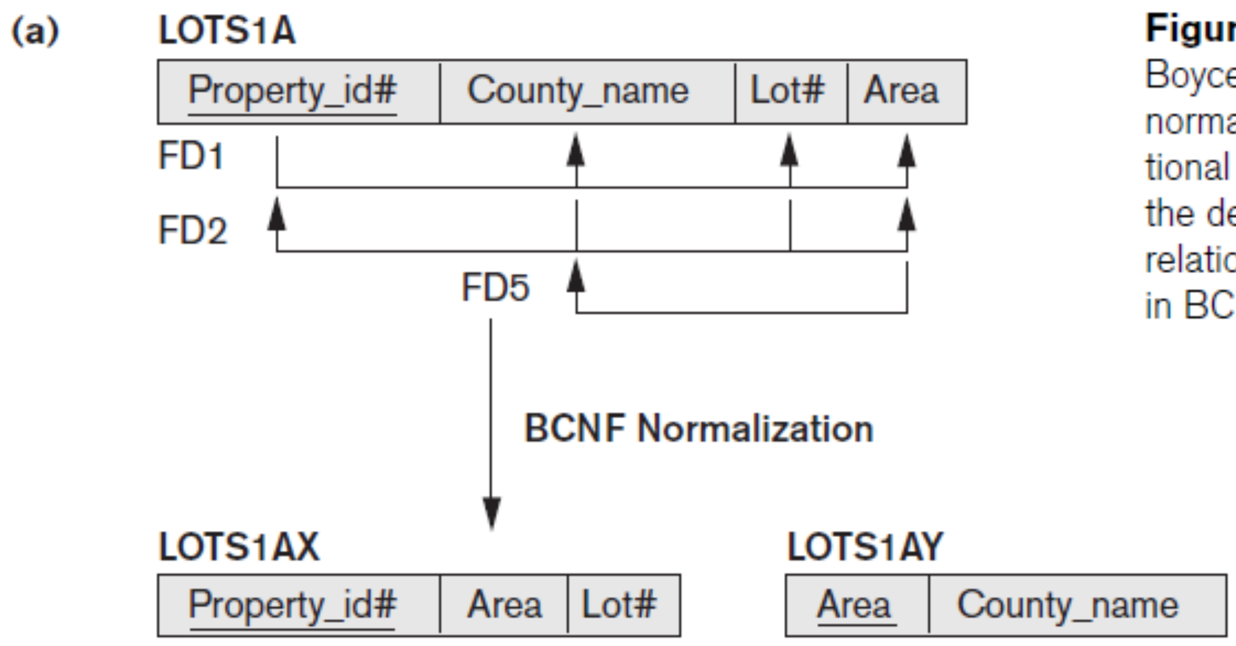


Figure 15.13
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.

Boyce-Codd Normal Form

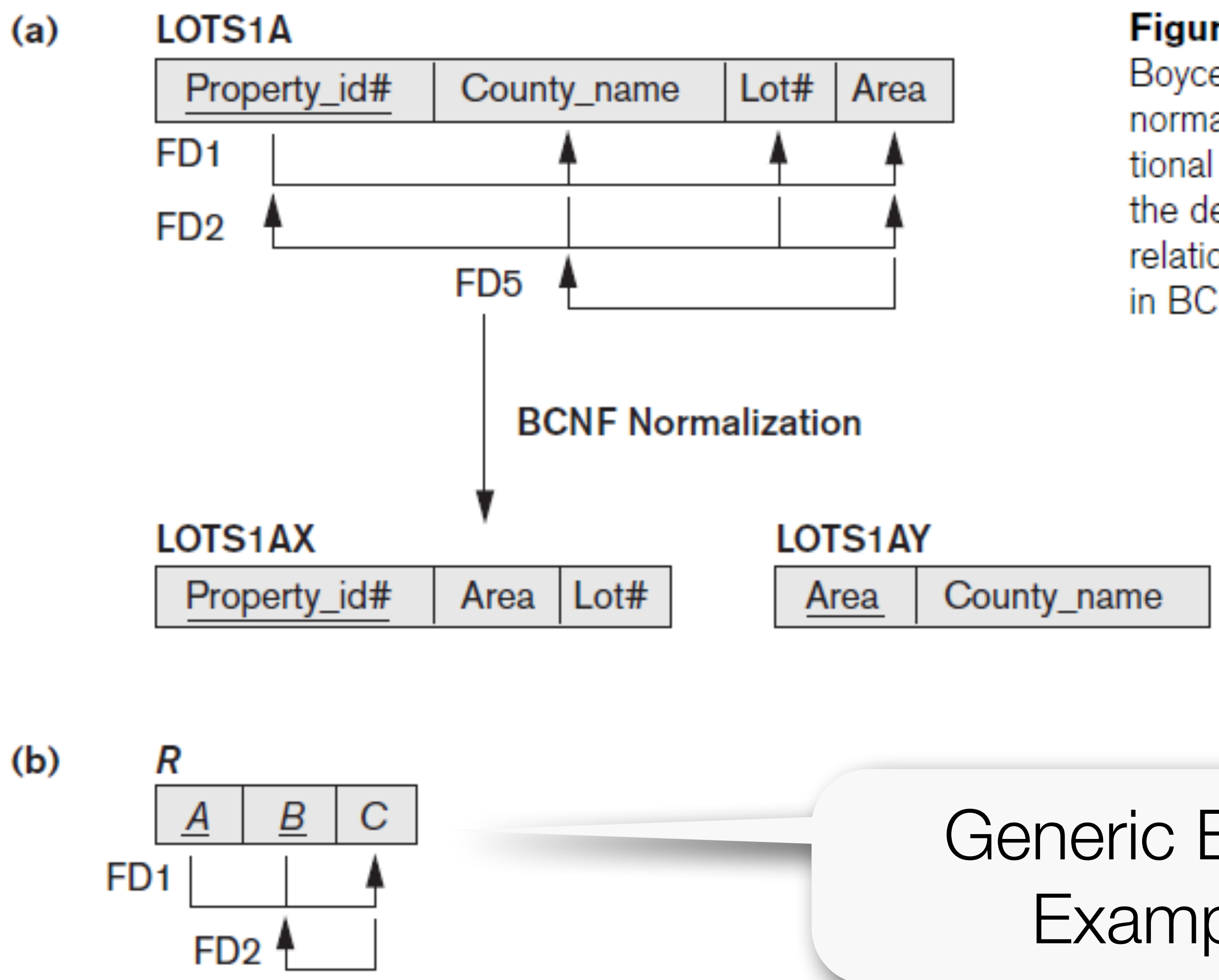


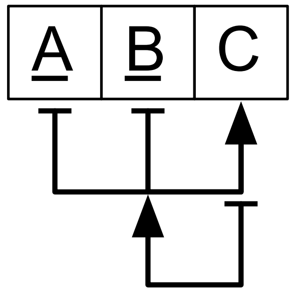
Figure 15.13
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.

Generic BCNF Example

BCNF

page 530-1

R



1. $\underline{A}\ \underline{B}$ $\quad$ $\underline{A}\ \underline{C}$

2. $\underline{A}\ \underline{B}$ $\quad$ $\underline{C}\ B$

3. $\underline{A}\ \underline{C}$ $\quad$ $\underline{C}\ B$

Boyce-Codd Normal Form Example

- As another example, consider Figure 15.14, which shows a relation TEACH with the following dependencies:
  - FD1: {Student, Course} → Instructor
  - FD2: Instructor → Course
- Note that {Student, Course} is a candidate key for this relation and that the dependencies shown follow the pattern in Figure 15.13(b), with Student as *A*, Course as *B*, and Instructor as *C*.

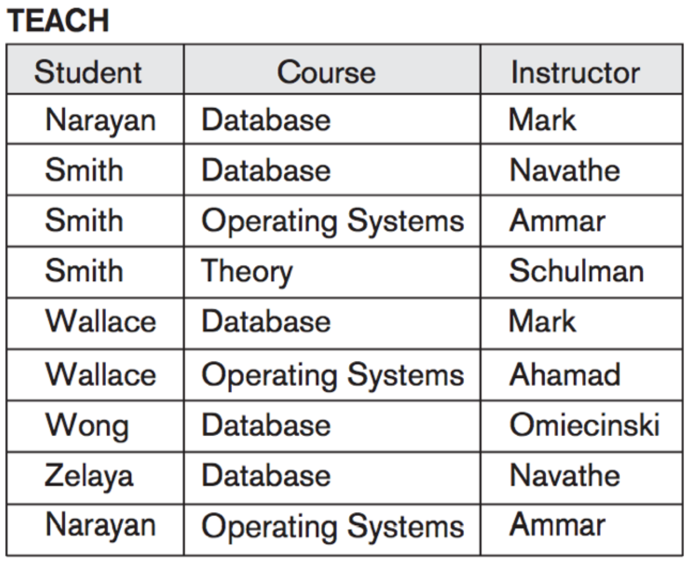


Figure 15.14
A relation TEACH that is in 3NF but not BCNF.

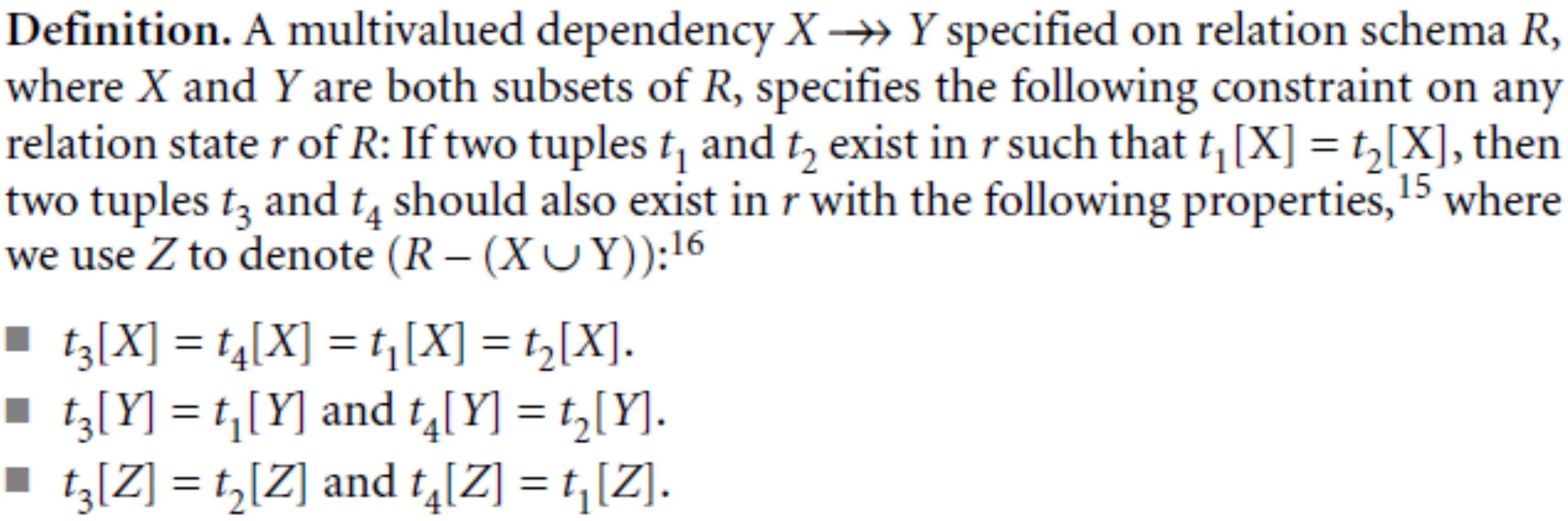
TEACH

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

Fourth Normal Form

Multivalued Dependency and Fourth Normal Form (4NF)

- Multivalued dependency (MVD)
  - Consequence of first normal form (1NF)
- Some relations have constraints that cannot be specified as functional dependencies.

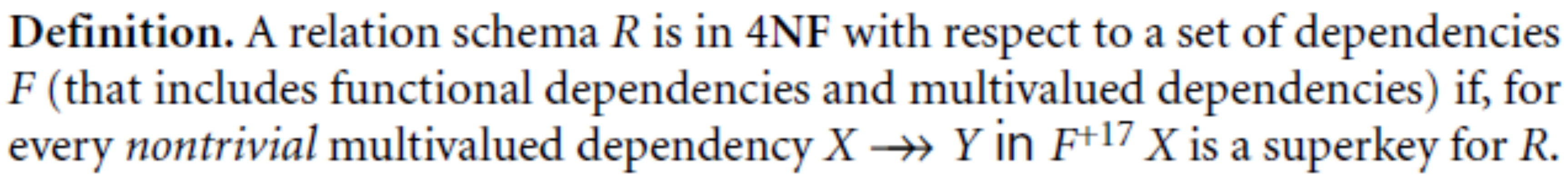


Definition. A multivalued dependency $X \twoheadrightarrow Y$ specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples t_1 and t_2 exist in r such that $t_1[X] = t_2[X]$, then two tuples t_3 and t_4 should also exist in r with the following properties,[15] where we use Z to denote $(R - (X \cup Y))$:[16]

- $t_3[X] = t_4[X] = t_1[X] = t_2[X]$.
- $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$.
- $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$.

Multivalued Dependency and Fourth Normal Form (cont'd.)

- Relations containing nontrivial MVDs
  - **All-key relations**
- **Fourth normal form (4NF)**
  - Violated when a relation has undesirable multivalued dependencies



Definition. A relation schema R is in **4NF** with respect to a set of dependencies F (that includes functional dependencies and multivalued dependencies) if, for every *nontrivial* multivalued dependency $X \twoheadrightarrow Y$ in F^{+}[17] X is a superkey for R.

Figure 15.15
Fourth and fifth normal forms.
(a) The EMP relation with two MVDs: Ename $\twoheadrightarrow$ Pname and Ename $\twoheadrightarrow$ Dname.
(b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

(a) **EMP**

Ename	Pname	Dname
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

(b) **EMP_PROJECTS**

Ename	Pname
Smith	X
Smith	Y

EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

Multivalued Dependency and Fourth Normal Form (cont'd.)

- For example, consider the relation EMP shown in Figure 15.15(a).
- A tuple in this EMP relation represents the fact that an employee whose name is Ename works on the project whose name is Pname and has a dependent whose name is Dname.
- An employee may work on several projects and may have several dependents, and the employee's projects and dependents are independent of one another.

Multivalued Dependency and Fourth Normal Form (cont'd.)

- To keep the relation state consistent, and to avoid any spurious relationship between the two independent attributes, we must have a separate tuple to represent every combination of an employee's dependent and an employee's project.
- This constraint is specified as a multivalued dependency on the EMP relation, which we define in this section.
- Informally, whenever two independent 1:*N* relationships *A*:*B* and *A*:*C* are mixed in the same relation, *R*(*A*, *B*, *C*), an MVD may arise.

Figure 15.15
Fourth and fifth normal forms.
(a) The EMP relation with two MVDs: Ename $\twoheadrightarrow$ Pname and Ename $\twoheadrightarrow$ Dname.
(b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

(a) **EMP**

Ename	Pname	Dname
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

(b) **EMP_PROJECTS**

Ename	Pname
Smith	X
Smith	Y

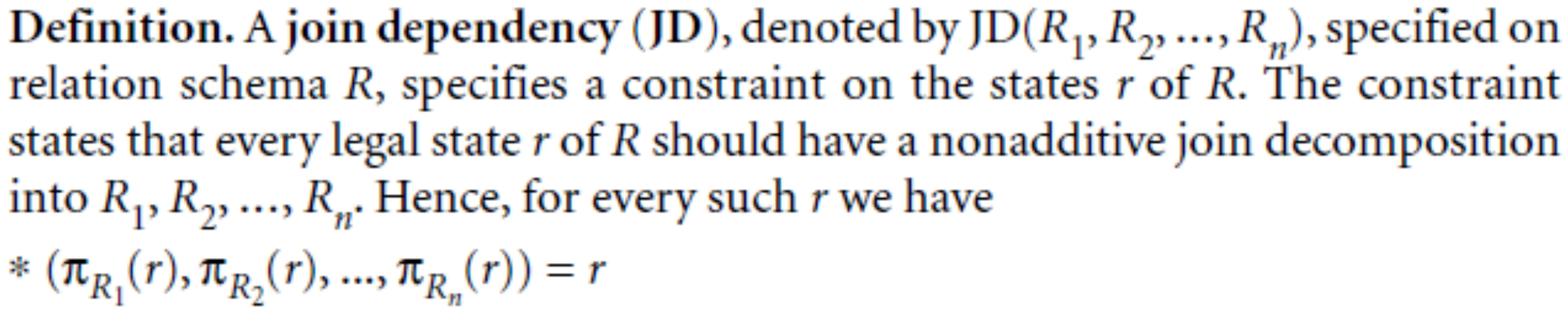
EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

Join Dependencies and Fifth Normal Form

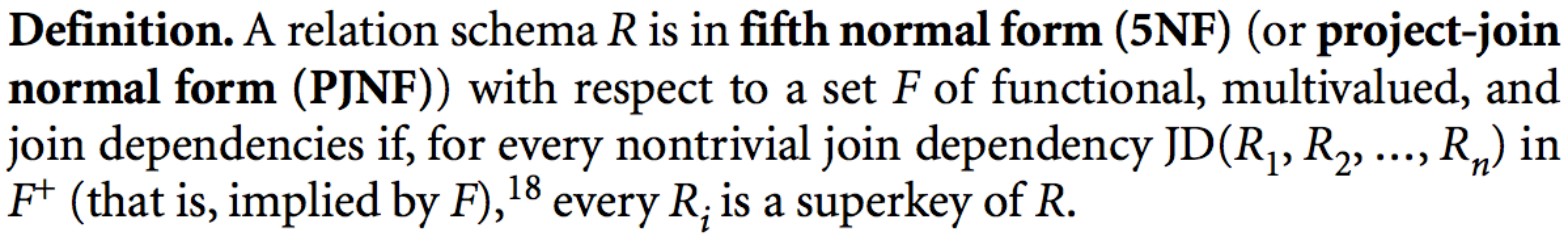
- **Join dependency**
- Multiway decomposition into fifth normal form (5NF)
- Very peculiar semantic constraint
  - Normalization into 5NF is very rarely done in practice

Join Dependencies and Fifth Normal Form (cont'd.)



Definition. A **join dependency** (JD), denoted by JD(R_1, R_2, ..., R_n), specified on relation schema R, specifies a constraint on the states r of R. The constraint states that every legal state r of R should have a nonadditive join decomposition into R_1, R_2, ..., R_n. Hence, for every such r we have

$$* (\pi_{R_1}(r), \pi_{R_2}(r), \ldots, \pi_{R_n}(r)) = r$$



Definition. A relation schema R is in **fifth normal form (5NF)** (or **project-join normal form (PJNF)**) with respect to a set F of functional, multivalued, and join dependencies if, for every nontrivial join dependency JD(R_1, R_2, ..., R_n) in F^+ (that is, implied by F),[18] every R_i is a superkey of R.

Fifth Normal Form (5NF)

(c) The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R_1, R_2, R_3).
(d) Decomposing the relation SUPPLY into the 5NF relations R_1, R_2, R_3.

(c) **SUPPLY**

Sname	Part_name	Proj_name
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

(d) ***R_1***

Sname	Part_name
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R_2

Sname	Proj_name
Smith	ProjX
Smith	ProjY
Adamsky	ProjY
Walton	ProjZ
Adamsky	ProjX

R_3

Part_name	Proj_name
Bolt	ProjX
Nut	ProjY
Bolt	ProjY
Nut	ProjZ
Nail	ProjX

Fifth Normal Form (5NF)

- For an example of a JD, consider once again the SUPPLY all-key relation in Figure 15.15(c).
- Suppose that the following additional constraint always holds: Whenever a supplier *s* supplies part *p*, and a project *j* uses part *p*, and the supplier *s* supplies at least one part to project *j*, then supplier *s* will also be supplying part *p* to project *j*.
- This constraint can be restated in other ways and specifies a join dependency JD(R_1, R_2, R_3) among the three projections R_1(Sname, Part_name), R_2(Sname, Proj_name), and R_3(Part_name, Proj_name) of SUPPLY.

Fifth Normal Form (5NF)

- If this constraint holds, the tuples below the dashed line in Figure 15.15(c) must exist in any legal state of the SUPPLY relation that also contains the tuples above the dashed line.
- Figure 15.15(d) shows how the SUPPLY relation with the join dependency is decomposed into three relations R_1, R_2, and R_3 that are each in 5NF.
- Notice that applying a natural join to any two of these relations produces spurious tuples, but applying a natural join to all three together does not.
- This is because only the JD exists, but no MVDs are specified. Notice, too, that the JD(R_1, R_2, R_3) is specified on all legal relation states, not just on the one shown in Figure 15.15(c).

Fifth Normal Form (5NF)

(c) The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R_1, R_2, R_3).
(d) Decomposing the relation SUPPLY into the 5NF relations R_1, R_2, R_3.

(c) SUPPLY

Sname	Part_name	Proj_name
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

(d) R_1

Sname	Part_name
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R_2

Sname	Proj_name
Smith	ProjX
Smith	ProjY
Adamsky	ProjY
Walton	ProjZ
Adamsky	ProjX

R_3

Part_name	Proj_name
Bolt	ProjX
Nut	ProjY
Bolt	ProjY
Nut	ProjZ
Nail	ProjX

Summary

- Informal guidelines for good design
- Functional dependency
  - Basic tool for analyzing relational schemas
- Normalization:
  - 1NF, 2NF, 3NF, BCNF, 4NF, 5NF