SOFTENG 351
Fundamentals of Database Systems

Basics of Functional Dependencies and Normalization for Relational Databases

Outline

- Informal Design Guidelines for Relational Databases
- Functional Dependencies (FDs)
- Normal Forms Based on Primary Keys
- General Normal Form Definitions for 2NF and 3NF (For Multiple Candidate Keys)
- BCNF (Boyce-Codd Normal Form)
- Multivalued Dependency and Fourth Normal Form
- Join Dependencies and Fifth Normal Form

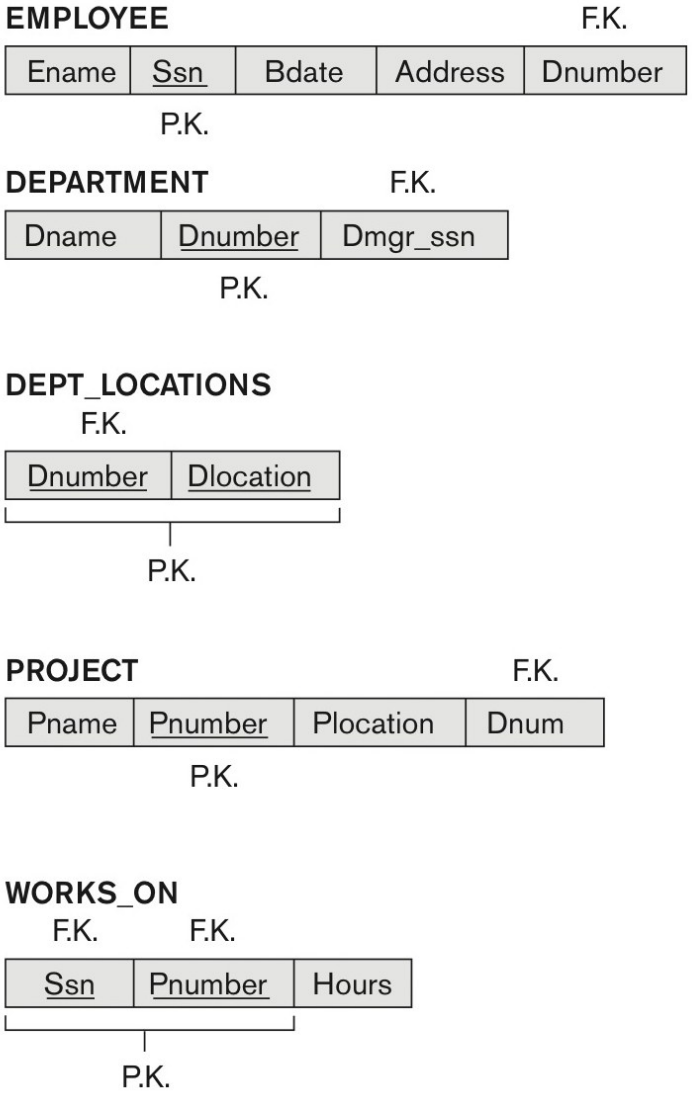
Informal Design Guidelines for Relational Databases

- 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?

Semantics of the Relational Attributes must be clear

- 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.*

A simplified COMPANY relational database schema and sample state



EMPLOYEE

Ename	Ssn	Bdate	Address	Dnumber
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1

DEPARTMENT

Dname	Dnumber	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

Ssn	Pnumber	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

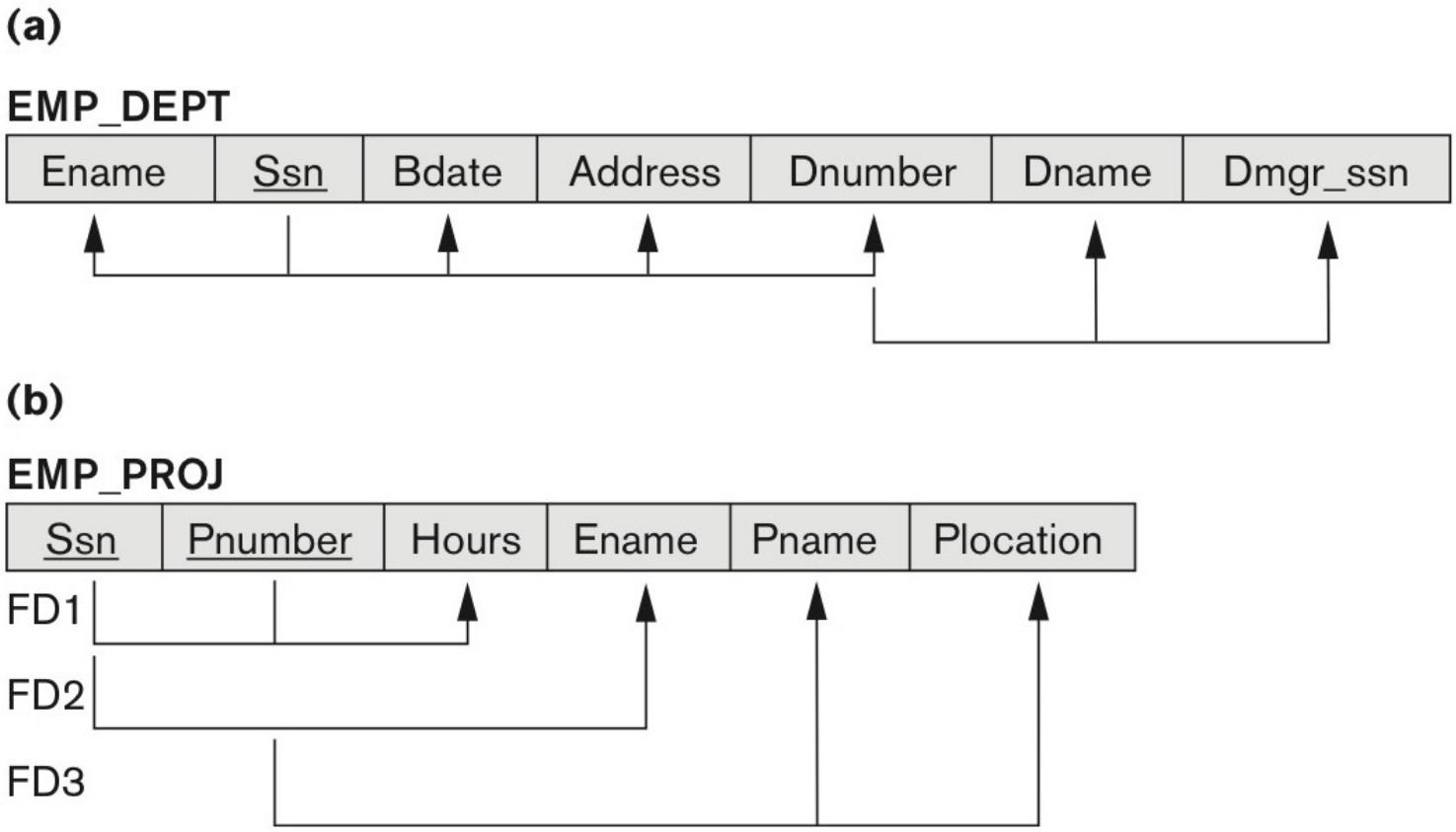
PROJECT

Pname	Pnumber	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

Redundant Information in Tuples and Update Anomalies

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

Two relation schemas suffering from update anomalies



EXAMPLE OF AN UPDATE ANOMALY

- Consider the relation:
  - EMP_PROJ(Emp#, Proj#, Ename, Pname, 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:
  - EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)
- Insert Anomaly:
  - Cannot insert a project unless an employee is assigned to it.
- Conversely
  - Cannot insert an employee unless an he/she is assigned to a project.

EXAMPLE OF A DELETE ANOMALY

- Consider the relation:
  - EMP_PROJ(Emp#, Proj#, Ename, Pname, 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.

Guideline for 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.

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

Generation of Spurious Tuples – avoid at any cost

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- GUIDELINE 4:
  - Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantee that no spurious tuples are generated.
  - Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples.

Summary of problematic relation schemas

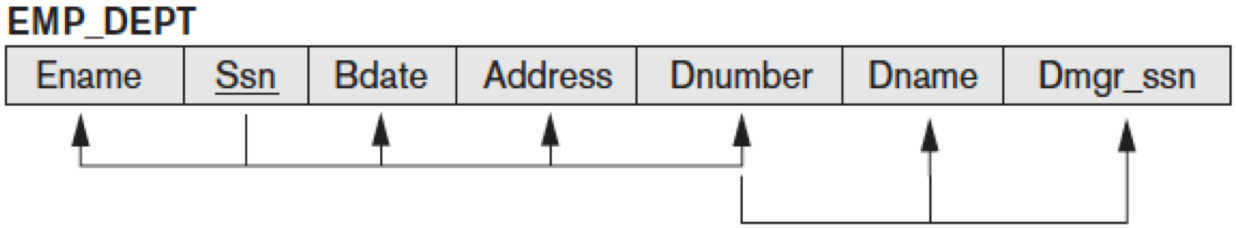
- Anomalies that cause redundant work to be done during insertion into and modification of a relation, and that may cause accidental loss of information during a deletion from a relation.
- Waste of storage space due to NULLs and the difficulty of performing selections, aggregation operations and joins due to NULL values
- Generation of invalid and spurious data during joins on base relations with matched attributes that may not represent a proper(foreign key, primary key) relationship.

Functional Dependencies

- Functional dependencies (FDs)
  - Are used to specify *formal measures* of the quality ("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

Defining Functional Dependencies

- X → Y holds if whenever two tuples have the same value for X, they *must have* the same value for Y
  - For any two tuples t1 and t2 in any relation instance r(R): If t1[X]=t2[X], *then* t1[Y]=t2[Y]
- X → Y in R specifies a *constraint* on all relation instances r(R)
- Written as X → Y, which can be displayed graphically on a relation schema as denoted by the arrow.



- FDs are derived from the real-world constraints on the attributes

Examples of FD constraints

- Social security number determines employee name
  - Ssn → Ename
- Project number determines project name and location
  - Pnumber → {Pname, Plocation}
- Employee ssn and project number determines the hours per week that the employee works on the project
  - {Ssn, Pnumber} → Hours

Examples of FD constraints

- A Functional Dependency (FD) is a property of the attributes in the relation 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]

Defining FDs from instances

- Note that in order to define the FDs, we need to understand the meaning of the attributes involved and the relationship between them.
- FD is a property of the attributes in the schema R
- Given the instance (population) of a relation, all we can conclude is that an FD *may exist* between certain attributes.
- What we can definitely conclude is – that certain FDs *do not exist* because there are tuples that show a violation of those dependencies.

Ruling Out FDs

Note that given the state of the TEACH relation, we can say that the FD: Text → Course may exist. However, the FDs Teacher → Course, Teacher → Text and Couse → Text are ruled out.

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

What FDs may exist?

- A relation *R*(A, B, C, D) with its extension.
- Which FDs *may exist* in this relation?

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

- FDs may hold for: B→C; C→B; {A,B}→C; {A,B}→D; {C,D}→B.

Normalization of Relations

- **Normalization of data:**
  - A process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desired properties of:
    - Minimizing redundancy
    - Minimizing the insertion, deletion and update anomalies
- **Normal form:**
  - Condition using keys and FDs of a relation to certify whether a relation schema is in a particular form
  - Refers to the highest normal form condition that it meets, indicate the degree of normalization

Normalization of Relations

- 2NF, 3NF, BCNF
  - based on keys and FDs of a relation schema
- 4NF
  - based on keys, multi-valued dependencies (MVD)s;
- 5NF
  - based on keys, join dependencies (JD)s
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation)

Additional Properties

- There are two important properties of normalization through decomposition:
  - **Non-additive join** (or ***lossless join***) property guarantees that the spurious tuples generation problem does not occur with respect to the relation schema after decomposition.
    - Is extremely important and *cannot* be sacrificed.
  - **Dependency preservation** property ensures that each functional dependency is represented in some individual relation resulting after the decomposition.
    - Is less stringent and may be sacrificed.

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 higher normal forms (e.g., 4NF and 5NF) 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 and BCNF, 4NF rarely used in practice.
  - Sometimes, relations may be left in lower normalization status (e.g., 2NF) for performance reasons.
- **Denormalization**:
  - 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

- A **superkey** of a relation schema R = {A1, A2, ..., An} is a set of attributes S *subset-of* R with the property that NO two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[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

- 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**.
- An attribute of relation R is called a **Prime attribute** of R, if it is a member of *some* candidate key of R.
- A **Nonprime attribute** is not a prime attribute – that is, it is not a member of any candidate key.

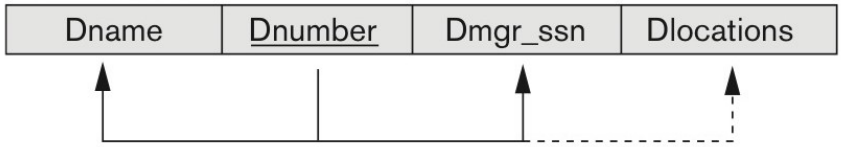
First Normal Form

- First Normal Form (1NF) is considered to be part of the formal definition of a relation in basic (flat) relational model. It was defined to disallow:
  - multivalued attributes
  - composite attributes
  - their combinations, i.e., **nested relations**: attributes whose values for an *individual tuple* are non-atomic
- The only attribute values permitted by 1NF are single atomic (or indivisible) values.
- Most RDBMSs allow only those relations to be defined that are in First Normal Form (1NF).

Normalization into 1NF

(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 14.9
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.

Normalizing into 1NF

- Three solutions to achieve 1NF of the previous example:
  - Remove the attribute Dlocation that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key Dnumber of the DEPARTMENT.
  - Expend the key to (Dnumber, Dlocation) so that there will be a separate tuple in the original DEPARTMENT relation for each location of a DEPARTMENT.
  - If a maximum number of values is known for the attribute, e.g., 3, replace the Dlocation attribute by the three atomic attributes: Dlocation1, Dlocation2 and Dlocation3.
- The first solution is the best, because it does not suffer from redundancy and it is more generic.

Second Normal Form

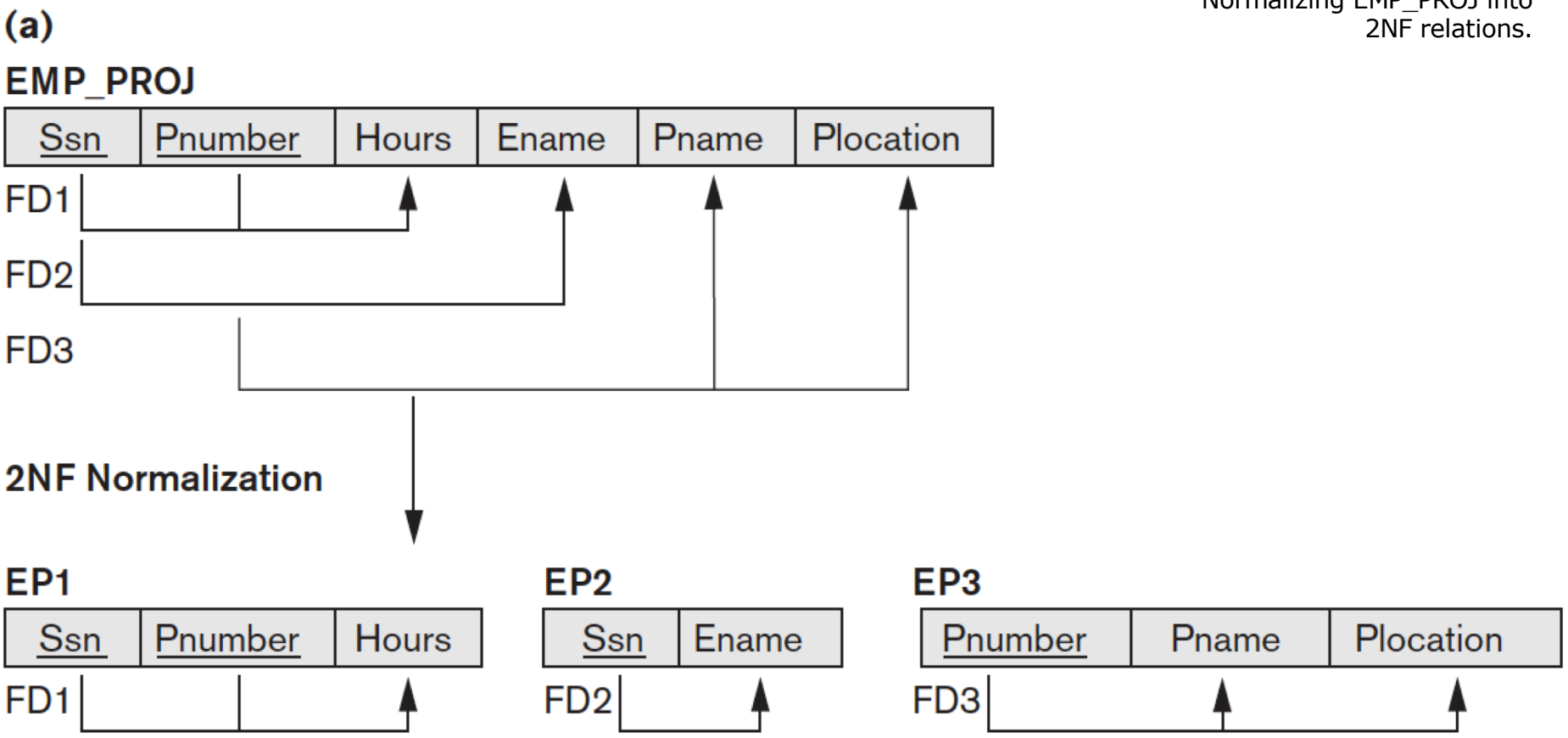
- Uses the concepts of **Functional Dependencies (FDs), primary key**
- Definitions
  - **Prime attribute:** An attribute that is member of the primary key K
  - **Full functional dependency:** a FD X → Y where removal of any attribute from X means that the FD does not hold any more
- Examples:
  - {Ssn, Pnumber} → Hours is a full FD since neither Ssn → Hours nor Pnumber → Hours hold
  - {Ssn, Pnumber} → Ename is not a full FD (it is called a partial dependency) since SSn → Ename also holds

Second Normal Form

- ***Definition***: 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
- The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key (or the full primary key).
- If a relation schema R is not in 2NF, it can be "second normalized" (or 2NF normalized) into a number of 2NF relations,
  - Nonprime attributes are associated only with the part of the original primary key on which they are fully functional dependent in the decomposed 2NF relations

Normalizing into 2NF

Normalizing into 2NF. (a) Normalizing EMP_PROJ into 2NF relations.



Third Normal Form

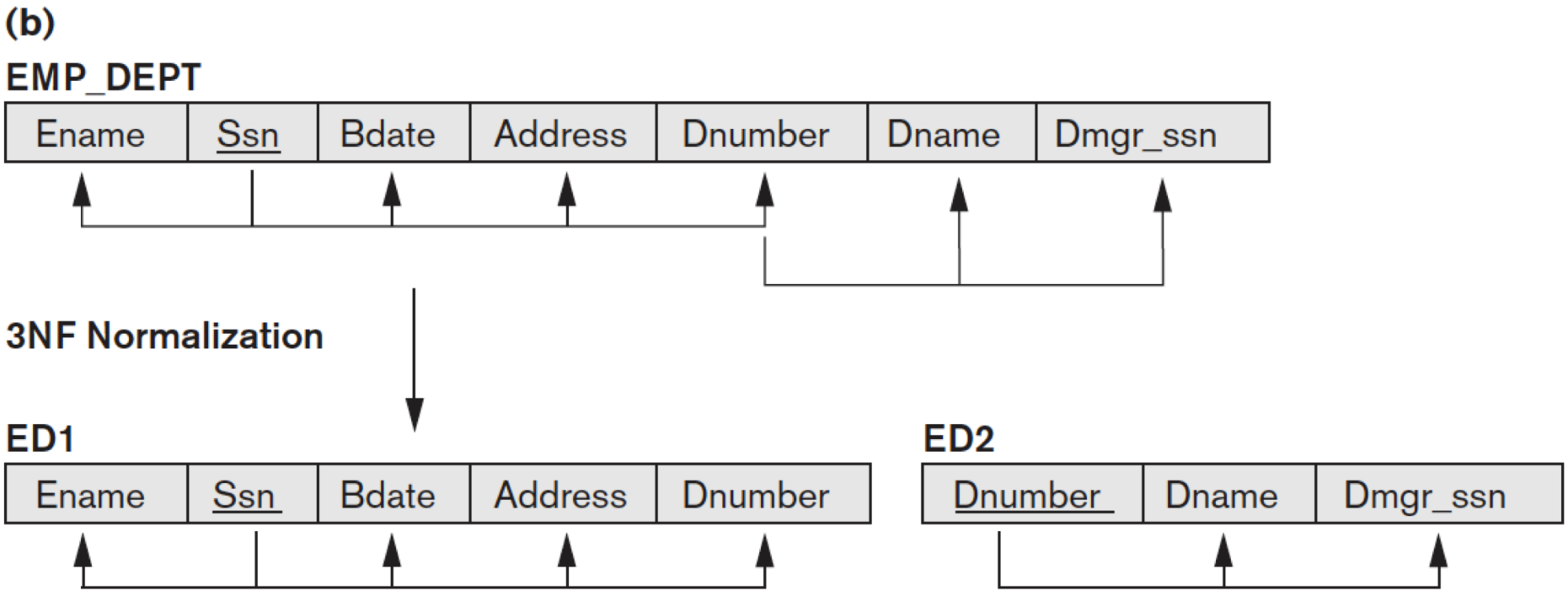
- A functional dependency X → Y in relation schema R is a **transitive dependency** if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R, and both X → Z and Z → Y hold.
- Examples:
  - Ssn → Dmgr_ssn is a **transitive** FD
    - Since Ssn → Dnumber and Dnumber → Dmgr_ssn hold
  - Ssn → Ename is **non-transitive**
    - Since there is no set of attributes X where Ssn → X and X → Ename

Third Normal Form

- ***Definition:*** A relation schema R is in **third normal form (3NF)** if it satisfies 2NF *and* has no non-prime attribute of R is transitively dependent on the primary key.
- R can be decomposed into 3NF relations via the process of 3NF normalization
- NOTE:
  - In X → Y and Y → 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 → Emp# → Salary and Emp# is a candidate key.

Normalizing into 3NF

Normalizing into 3NF.
(b) Normalizing EMP_DEPT into 3NF relations.



Summary of Normal Forms and their Normalization

Table 14.1 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 Normal Form Definitions (For Multiple Keys)

- Normal forms defined informally
  - 1NF: All attributes depend on **the key**
  - 2NF: All attributes depend on **the whole key**
  - 3NF: All attributes depend on **nothing but the key**
- The above definitions consider the primary key only
- The following more general definitions take into account relations with multiple candidate keys
- Any attribute involved in a candidate key is a *prime attribute*
- All other attributes are called *non-prime attributes.*

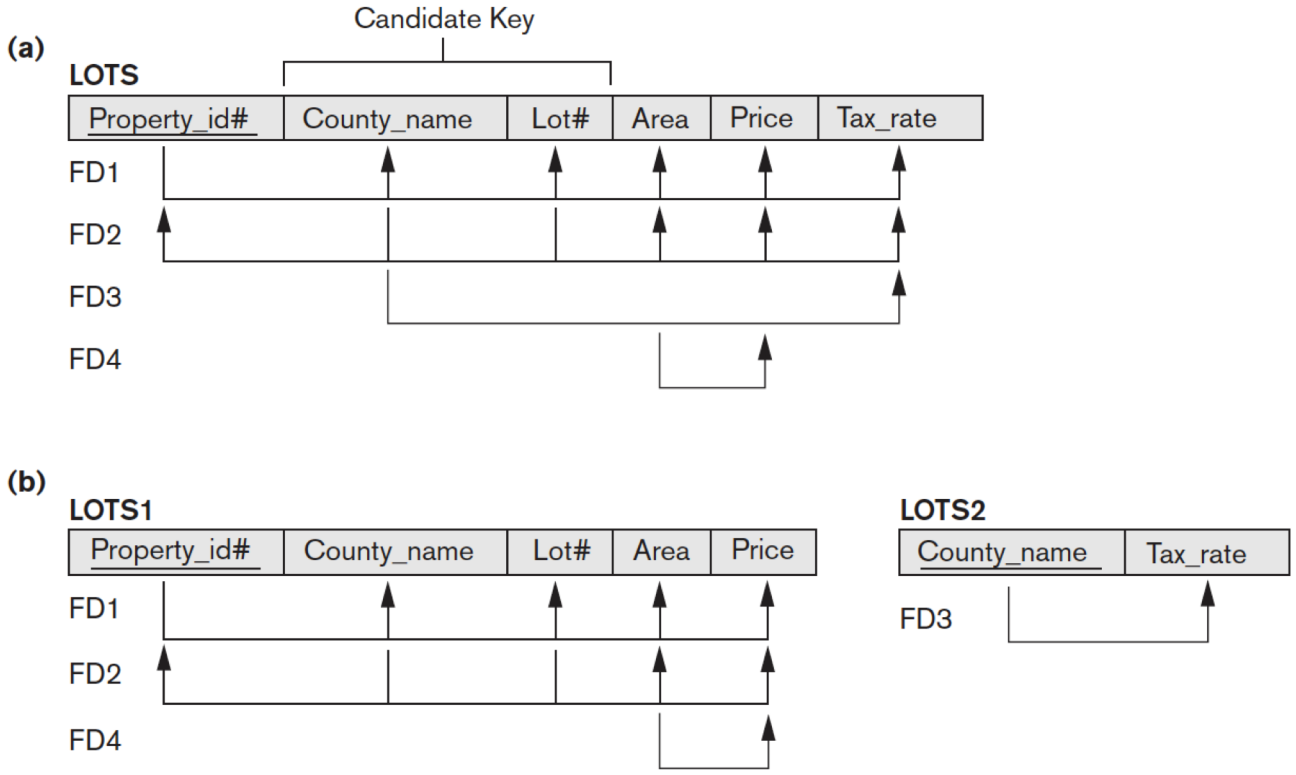
General Definition of 2NF (For Multiple Candidate Keys)

- 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

The FD County_name → Tax_rate violates 2NF.

So second normalization converts LOTS into:

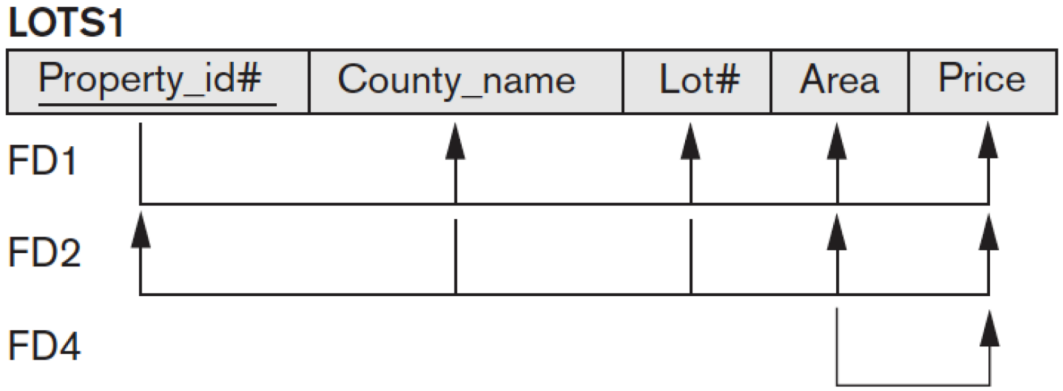
LOTS1 (Property_id#, County_name, Lot#, Area, Price)
LOTS2 (County_name, Tax_rate)



General Definition of Third Normal Form

- **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 nontrivial functional dependency X → A holds in R, then either:
  a) X is a superkey of R, **or**
  b) A is a prime attribute of R

LOTS1 relation violates 3NF because Area → Price; and Area is not a superkey in LOTS1.



Interpreting the General Definition of Third Normal Form

- A relation schema R violates the general definition of **3NF** if whenever a FD X → A holds in R that meets either of the two conditions (a) X is a superkey of R and (b) A is a prime attribute of R
- Condition (a) catches two types of violations :
  1. A non-prime attribute functionally determines another non-prime attribute. This catches 3NF violations due to a transitive dependency.
  2. A proper subset of a key of R functionally determines a non-prime attribute. This catches 2NF violations due to non-full functional dependencies.

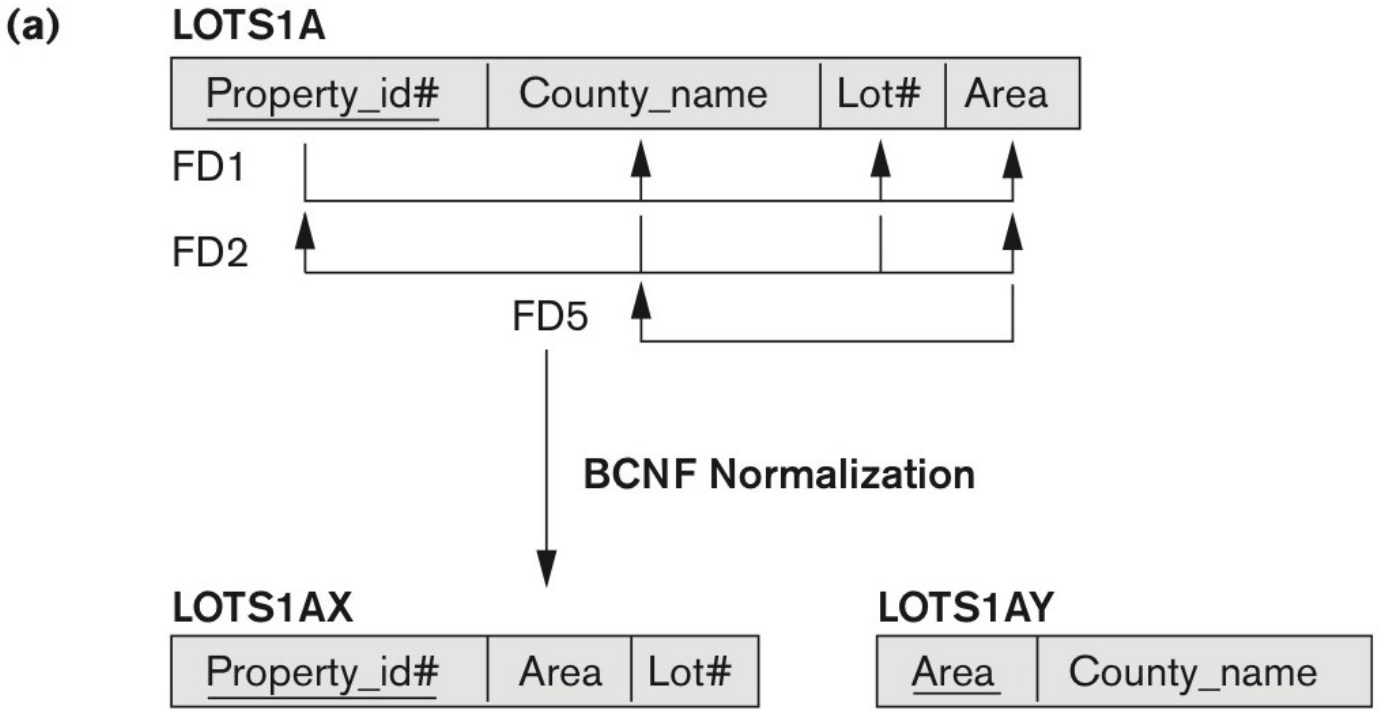
Interpreting the General Definition of Third Normal Form

- **ALTERNATIVE DEFINITION of 3NF:** We can restate the definition as:
  - A relation schema R is in **third normal form (3NF)** if every non-prime attribute in R meets both of these conditions:
  - It is fully functionally dependent on every key of R
  - It is non-transitively dependent on every key of R
  - Note that stated this way, a relation in 3NF also meets the requirements for 2NF.
- The condition (b) from the previous slide takes care of the dependencies that "slip through" (are allowable to) 3NF but are "caught by" BCNF which we discuss next.

Boyce-Codd Normal Form (BCNF)

- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever a nontrivial functional dependency **X → A** holds in R, then **X is a superkey** of R.
- Each normal form is strictly stronger than the previous one, i.e.,
  - 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
- Hence BCNF is considered a stronger form of 3NF
- The goal is to have each relation in BCNF (or 3NF)

Boyce-Codd normal form



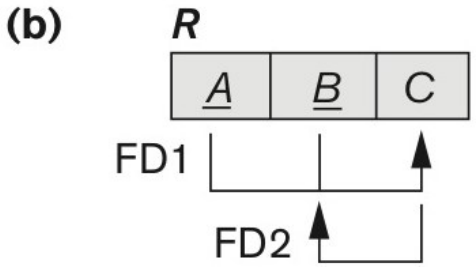


Figure 14.13
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF due to the FD: C → B.

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

- Two FDs exist in the relation TEACH:
  - FD1: {Student, Course} → Instructor
  - FD2: Instructor → Course
- {Student, Course} is a candidate key for this relation
- So the 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.

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

Achieving the BCNF by Decomposition

- Three possible decompositions for relation TEACH
  - D1: {Student, Instructor} and {Student, Course}
  - D2: {Course, Instructor} and {Course, Student}
  - D3: {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 (lossless) 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).

Test for checking non-additivity of Binary Relational Decompositions

- Testing Binary Decompositions for **Lossless Join (Non-additive Join)** Property
  - **Binary Decomposition:** Decomposition of a relation R into two relations.
  - **PROPERTY NJB (non-additive join test for binary decompositions):** A decomposition D = {R1, R2} 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: ((R1 ∩ R2) → (R1- R2)) is in F^+, **or**
    - The FD: ((R1 ∩ R2) → (R2 - R1)) is in F^+.
      - The notion of F^+ (call **closure** of F) refers to the set of all functional dependencies that include F as well as all dependencies that can be inferred from F.

Test for checking non-additivity of Binary Relational Decompositions

- If you apply the **NJB** test to the 3 decompositions of the TEACH relation:
  - D1 gives **Student** → Instructor or **Student** → Course, none of which is true.
  - D2 gives **Course** → Instructor or **Course** → Student, none of which is true.
  - However, in D3 we get **Instructor** → Course or **Instructor** → Student.
- Since **Instructor** → Course is indeed true, the NJB property is satisfied and D3 is determined as a non-additive (good) decomposition.

Multivalued Dependencies and Fourth Normal Form

- **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 $t_1[X] = t_2[X]$, two tuples t_3 and t_4 should also exist in r with the following properties, where Z is used to denote $(R - (X \cup Y))$:
  - $t_3[X] = t_4[X] = t_1[X] = t_2[X]$.
  - $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$.
  - $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$.
- An MVD $X \twoheadrightarrow Y$ in R is called a **trivial MVD** if (a) Y is a subset of X, or (b) $X \cup Y = R$.

Multivalued Dependencies and Fourth Normal Form

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^+, *X* is a superkey for R.
- 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*.

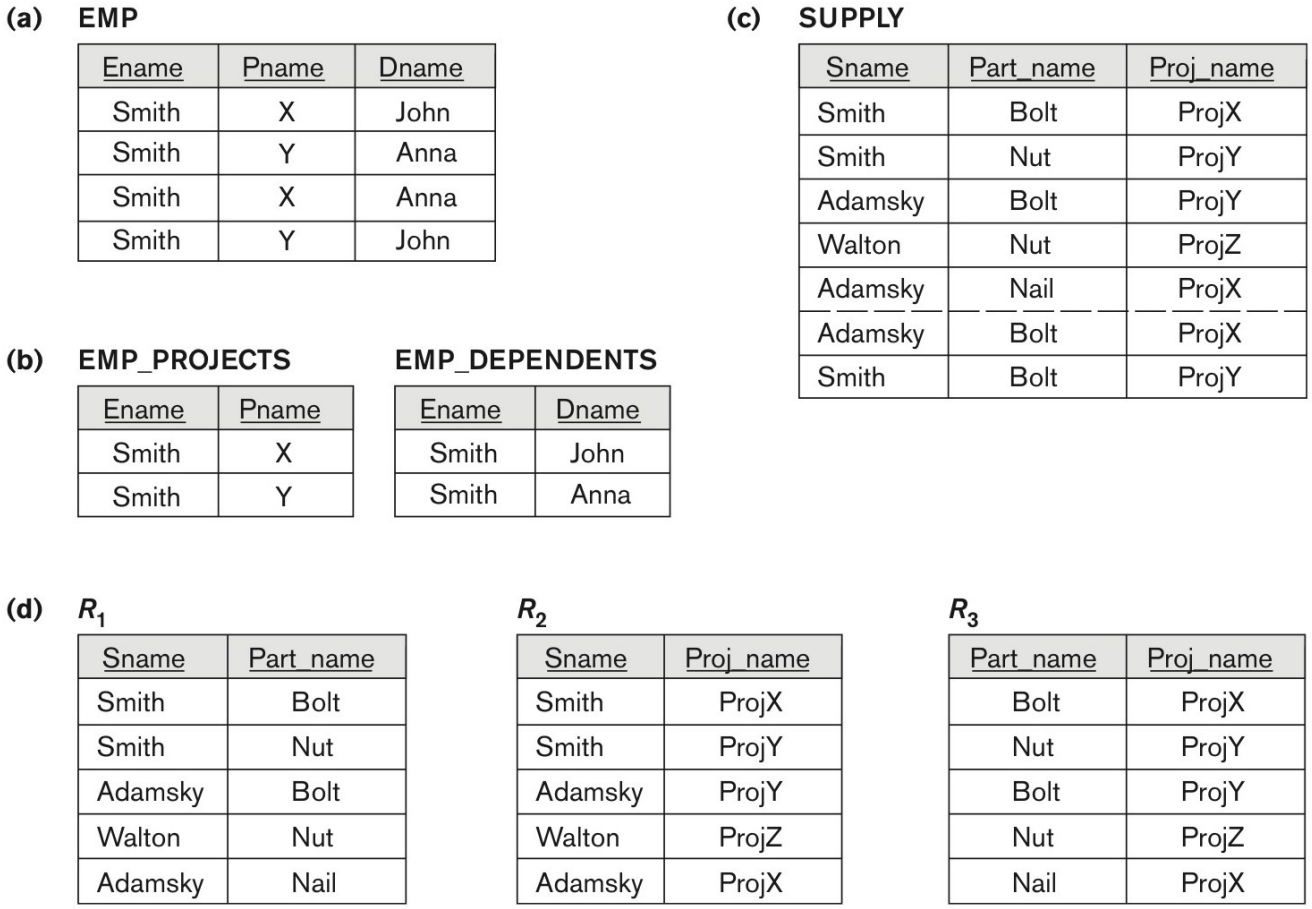
Join Dependencies and Fifth Normal Form

- 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 non-additive join decomposition into R_1, R_2, ..., R_n; that is, for every such r we have
  - * ($\pi_{R1}(r)$, $\pi_{R2}(r)$, ..., $\pi_{Rn}(r)$) = r
  - ***Note**: an MVD is a special case of a JD where $n = 2$.*
- A join dependency JD(R_1, R_2, ..., R_n), specified on relation schema R, is a **trivial JD** if one of the relation schemas R_i in JD(R_1, R_2, ..., R_n) is equal to R.

Join Dependencies and Fifth Normal Form

- 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*),
  - every R_i is a superkey of *R*.
- Discovering join dependencies in practical databases with hundreds of relations is next to impossible. Therefore, 5NF is rarely used in practice.

Fourth and fifth normal forms



(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

(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

Figure 14.15
Fourth and fifth normal forms. (a) The EMP relation with two MVDs: Ename ↠ Pname and Ename ↠ Dname. (b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS. (c) The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R1, R2, R3). (d) Decomposing the relation SUPPLY into the 5NF relations R1, R2, R3.

Summary

- Informal Design Guidelines for Relational Databases
- Functional Dependencies (FDs)
- Normal Forms (1NF, 2NF, 3NF) – based on Primary Keys
- General Normal Form Definitions of 2NF and 3NF (For Multiple Keys)
- BCNF (Boyce-Codd Normal Form)
- Fourth and Fifth Normal Forms