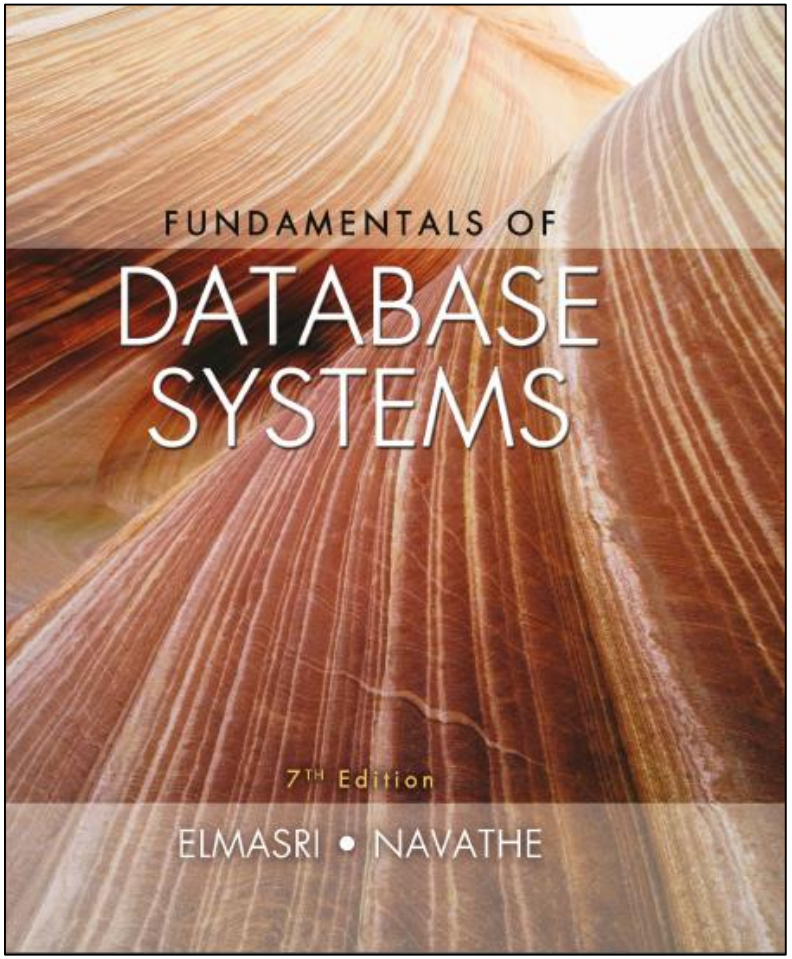
Fundamentals of Database Systems

Seventh Edition



Chapter 14

Basics of Functional Dependencies and Normalization for Relational Databases

Learning Objectives

14.1 Informal Design Guidelines for Relational Databases

- **14.1.1** Semantics of the Relation Attributes
- **14.1.2** Redundant Information in Tuples and Update Anomalies
- **14.1.3** Null Values in Tuples
- **14.1.4** Spurious Tuples

14.2 Functional Dependencies (FDs)

- **14.2.1** Definition of Functional Dependency

Learning Objectives

14.3 Normal Forms Based on Primary Keys

- **14.3.1** Normalization of Relations
- **14.3.2** Practical Use of Normal Forms
- **14.3.3** Definitions of Keys and Attributes Participating in Keys
- **14.3.4** First Normal Form
- **14.3.5** Second Normal Form
- **14.3.6** Third Normal Form

14.4 General Normal Form Definitions for 2NF and 3NF (For Multiple Candidate Keys)

14.5 BCNF (Boyce-Codd Normal Form)

14.1 Informal Design Guidelines for Relational Databases (1 of 2)

- 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?

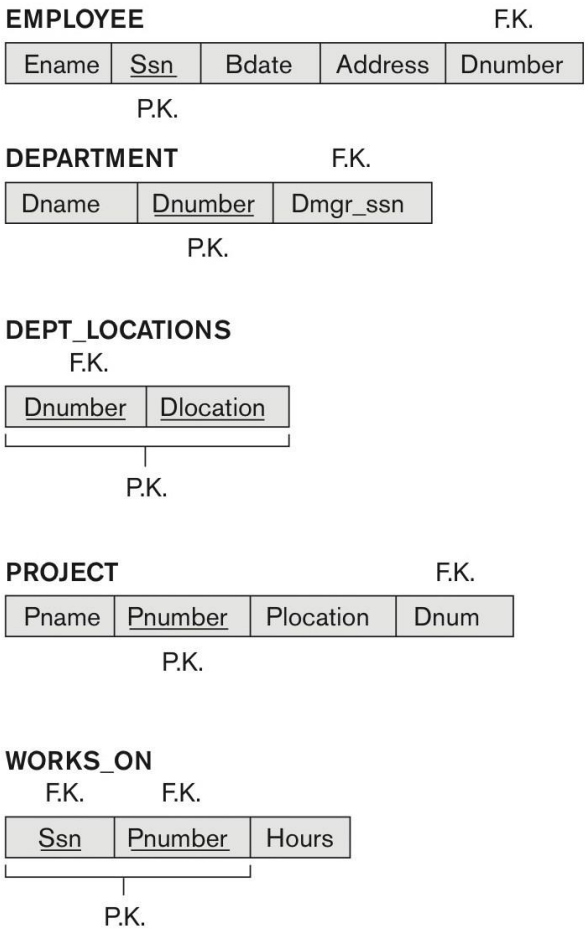
14.1 Informal Design Guidelines for Relational Databases (2 of 2)

- We first discuss informal guidelines for good relational design
- Then we discuss formal concepts of functional dependencies and normal forms
  - 1NF (First Normal Form)
  - 2NF (Second Normal Form)
  - 3NF (Third Noferferferfewrmal Form)
  - BCNF (Boyce-Codd Normal Form)

14.1.1 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.

Figure 14.1 A Simplified COMPANY Relational Database Schema



14.1.2 Redundant Information in Tuples and Update Anomalies

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

Example of an Update Anomaly

- Consider the relation:
  - 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.

Figure 14.3 Two Relation Schemas Suffering from Update Anomalies

(a) EMP_DEPT and (b) EMP_PROJ

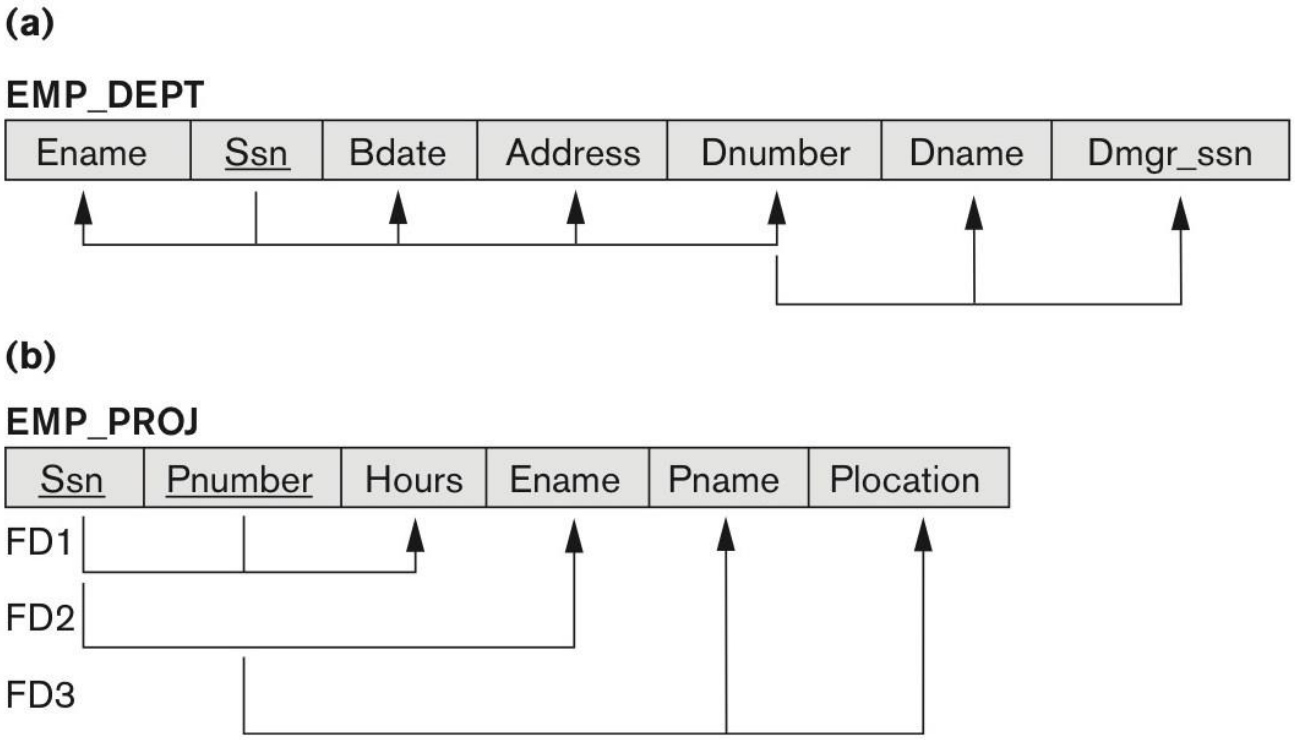


Figure 14.4 Sample States For EMP_DEPT and EMP_PROJ

Sample states for EMP_DEPT and EMP_PROJ resulting from applying NATURAL JOIN to the relations in Figure 14.2. These may be stored as base relations for performance reasons.

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

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 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.

14.1.3 Null Values in Tuples

- **Guideline 3:**
  - Relations should be designed such that their tuples will have as few NULL values as possible
  - Attributes that are NULL frequently could be placed in separate relations (with the primary key)
- Reasons for nulls:
  - Attribute not applicable or invalid
  - Attribute value unknown (may exist)
  - Value known to exist, but unavailable

14.1.4 Generation of Spurious Tuples – Avoid at Any Cost (1 of 2)

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- The “lossless join” property is used to guarantee meaningful results for join operations
- **Guideline 4:**
  - The relations should be designed to satisfy the lossless join condition.
  - No spurious tuples should be generated by doing a natural-join of any relations.

14.1.4 Generation of Spurious Tuples – Avoid at Any Cost (2 of 2)

- There are two important properties of decompositions:

  a) Non-additive or losslessness of the corresponding join

  b) Preservation of the functional dependencies.

- Note that:
  - Property (a) is extremely important and **cannot** be sacrificed.
  - Property (b) is less stringent and may be sacrificed. (See Chapter 15).

Summary of Design Guidelines

- 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

Learning Objectives

14.1 Informal Design Guidelines for Relational Databases

- **14.1.1** Semantics of the Relation Attributes
- **14.1.2** Redundant Information in Tuples and Update Anomalies
- **14.1.3** Null Values in Tuples
- **14.1.4** Spurious Tuples

14.2 Functional Dependencies (FDs)

- **14.2.1** Definition of Functional Dependency

14.2 Functional Dependencies

- Functional dependencies (FDs)
  - Are used to specify **formal measures** of the "goodness" of relational designs
  - And keys are used to define **normal forms** for relations
  - Are **constraints** that are derived from the **meaning** and **interrelationships** of the data attributes
- A set of attributes *X* **functionally determines** a set of attributes *Y* if the value of *X* determines a unique value for *Y*

14.2.1 Defining Functional Dependencies

- $X \rightarrow Y$ holds if whenever two tuples have the same value for X, they **must have** the same value for Y
  - For any two tuples t_1 and t_2 in any relation instance $r(R)$: If $t_1[X] = t_2[X]$, **then** $t_1[Y] = t_2[Y]$
- $X \rightarrow Y$ in R specifies a **constraint** on all relation instances $r(R)$
- Written as $X \rightarrow Y$ can be displayed graphically on a relation schema as in Figures. (denoted by the arrow:).
- FDs are derived from the real-world constraints on the attributes

Examples of FD Constraints (1 of 2)

- Social security number determines employee name
  - SSN → ENAME
- Project number determines project name and location
  - PNUMBER → {PNAME, PLOCATION}

Examples of FD Constraints (1 of 2)

- Employee ssn and project number determines the hours per week that the employee works on the project
  - {SSN, PNUMBER} → HOURS

Essn	Pno	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

Examples of FD Constraints (2 of 2)

- An FD is a property of the attributes in the schema *R*
- The constraint must hold on **every** relation instance *r*(*R*)
- If *K* is a key of *R*, then *K* functionally determines all attributes in *R*
  - (since we never have two distinct tuples with $t_1[K]=t_2[K]$)

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.
- An 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.

Figure 14.7 Ruling Out FDs

Note that given the state of the TEACH relation, we can say that the **FD: Text → Course** may exist.

However, the FD:

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

Figure 14.8 What FDs May Exist?

- A relation 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

What FDs May Exist?

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

STUD_NO	STUD_NAME	STUD_PHONE	STATE	COUNTRY	STUD-AGE
1	John	6096271721	NJ	USA	20
2	Marry	7218291281	NJ	USA	19
3	Ann	2128291981	NY	USA	18
4	Sue	NULL	NY	USA	21
5	Eren	NULL	Izmir	Turkey	25

What FDs May Exist?

A relation STUDENT(SS#,NAME,MAJOR,COURSE,GRADE) with its instances. Which FDs may exist in this relation?

SS#	NAME	MAJOR	COURSE	GRADE
123	TOM	CIS	CIS 3500	A
123	TOM	CIS	CIS 4500	B
123	TOM	CIS	CIS 4300	A
200	AMY	FIN	ACC 2120	B
200	AMY	FIN	FIN 2000	B
200	AMY	FIN	CIS 2000	A
300	BILL	CIS	CIS 4300	A

Functional Dependencies

The attributes listed on the left hand side of the -> are called determinants.

One can read A -> B as, **"A determines B".**

A **key** functionally determines a tuple (row).

Not all *determinants* are keys.

Definitions of Keys and Attributes Participating in Keys

Superkey

Key

Candidate key

Prime attribute

Nonprime attribute

Full functional dependency

Partial dependency

Transitive functional dependency

Definitions of Keys and Attributes Participating in Keys

- A **superkey** of a relation schema is $R = \{A_1, A_2, ..., A_n\}$

  a set of attributes S **subset-of** *R* with the property that no two tuples t_1 and t_2 in any legal relation state *r* of *R* will have $t_1[S] = t_2[S]$

  A superkey is a combination of columns that uniquely identifies any row within a relational database management system (RDBMS) table.

Definitions of Keys and Attributes Participating in Keys

EMPLOYEE

Emp_SSN	Emp_Number	Emp_Name
123456789	226	Steve
999999321	227	Ajeet
888997212	228	Chaitanya
777778888	229	Steve

super keys: uniquely identify a row of the employee table

Definitions of Keys and Attributes Participating in Keys

EMPLOYEE

Emp_SSN	Emp_Number	Emp_Name
123456789	226	Steve
999999321	227	Ajeet
888997212	228	Chaitanya
777778888	229	Steve

The above table has following super keys. All of the following sets of super key are able to uniquely identify a row of the employee table.

{Emp_SSN}

Definitions of Keys and Attributes Participating in Keys

- 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.

{Emp_SSN}

{Emp_Number}

{Emp_SSN, Emp_Number}

{Emp_SSN, Emp_Name} ←

{Emp_Number, Emp_Name} ←

{Emp_SSN, Emp_Number, Emp_Name} ←

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**.

A candidate key is a closely related concept of superkey where the superkey is reduced to the minimum number of columns required to uniquely identify each row.

{{Emp_SSN} , {Emp_Number} } candidate keys

{Emp_SSN} primary key

{Emp_Number} secondary key

Definitions of Keys and Attributes Participating in Keys

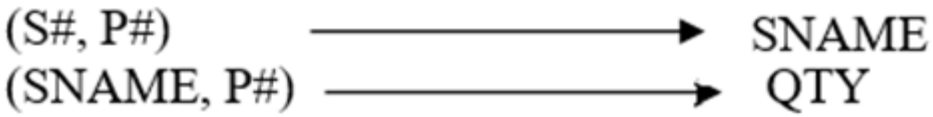
- A **Prime attribute** must be a member of **some** candidate key

{Emp_SSN}

- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

{Emp_Name}

Partial / Full Functional dependency



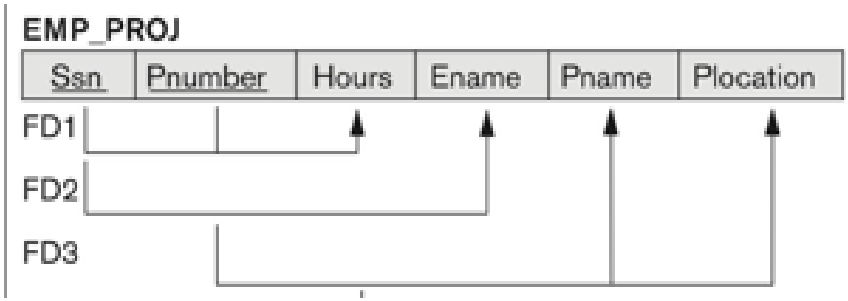
Partial Functional dependency:
Only **ONE attribute** in the left hand side is sufficient for this dependency.

(S#, P#) ⟶ SNAME

Full Functional dependency:
BOTH attributes are required in the left hand side of this functional dependency.

(S#, P#) ⟶ QTY

Partial / Full Functional dependency



{SSN, PNUMBER} → ENAME

is not a full FD (it is called a **partial dependency**) since

SSN → ENAME also holds

{SSN, PNUMBER} → HOURS
is a **full FD** since neither

SSN → HOURS nor

PNUMBER → HOURS hold

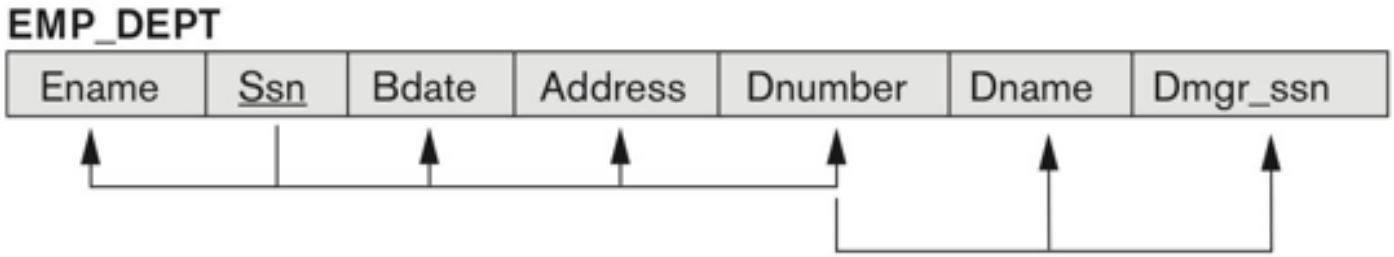
Transitive functional dependency

Transitive functional dependency:

a FD $X \rightarrow Z$ that can be derived from two FDs

$X \rightarrow Y$ and $Y \rightarrow Z$

Transitive functional dependency

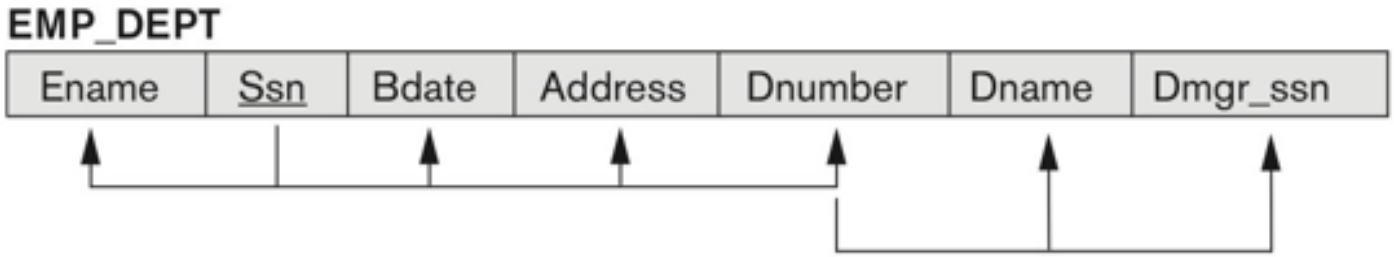


SSN → {ENAME, BDATE, ADDRSS, DNUMBER}

DNUMBER → {DNAME, DMGR_SSN}

SSN → DNAME

Transitive functional dependency



SSN → DMGRSSN is a **transitive** FD

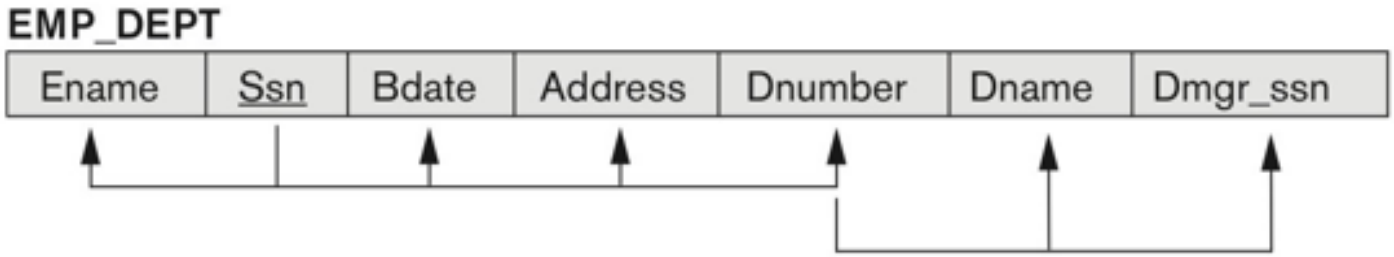
Since SSN → DNUMBER and DNUMBER → DMGRSSN hold

SSN → ENAME is **non-transitive**

Since there is no set of attributes *X* where

SSN → *X* and *X* → ENAME

Transitive functional dependency



Primary Attribute: SSN

Non-Primary Attributes: {ENAME, BDATE, ADDRESS, DNUMBER, DNAME, DMGR_SSN}

Learning Objectives

14.3 Normal Forms Based on Primary Keys

- **14.3.1** Normalization of Relations
- **14.3.2** Practical Use of Normal Forms
- **14.3.3** Definitions of Keys and Attributes Participating in Keys
- **14.3.4** First Normal Form
- **14.3.5** Second Normal Form
- **14.3.6** Third Normal Form

14.4 General Normal Form Definitions for 2NF and 3NF (For Multiple Candidate Keys)

14.5 BCNF (Boyce-Codd Normal Form)

14.3.1 Normalization of Relations (1 of 2)

- **Normalization:**
  - The process of decomposing unsatisfactory “bad” relations by breaking up their attributes into smaller relations
- **Normal form:**
  - Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

14.3.1 Normalization of Relations (2 of 2)

- 2NF, 3NF, BCNF
  - based on keys and FDs of a relation schema
- 4NF
  - based on keys, multi-valued dependencies: MVDs;
- 5NF
  - based on keys, join dependencies: JDs
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation; see Chapter 15)

14.3.2 Practical Use of Normal Forms

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are **hard to understand** or to **detect**
- The database designers **need not** normalize to the highest possible normal form
  - (usually up to 3NF and BCNF. 4NF rarely used in practice.)
- **Denormalization**:
  - The process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

14.3.4 First Normal Form

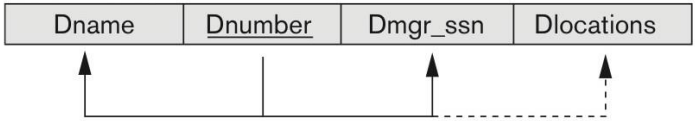
- Disallows
  - composite attributes
  - multivalued attributes
  - **nested relations**; attributes whose values for an **individual tuple** are non-atomic
- Considered to be part of the definition of a relation
- Most RDBMSs allow only those relations to be defined that are in First Normal Form

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.

(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 14.10 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.

(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

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

14.3.5 Second Normal Form (1 of 2)

- Uses the concepts of **FDs, primary key**
- Definitions
  - **Prime attribute**: An attribute that is member of the primary key K
  - **Full functional dependency**: a FD Y → Z where removal of any attribute from Y means 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

14.3.5 Second Normal Form (2 of 2)

- A relation schema *R* is in second normal form (2NF) if

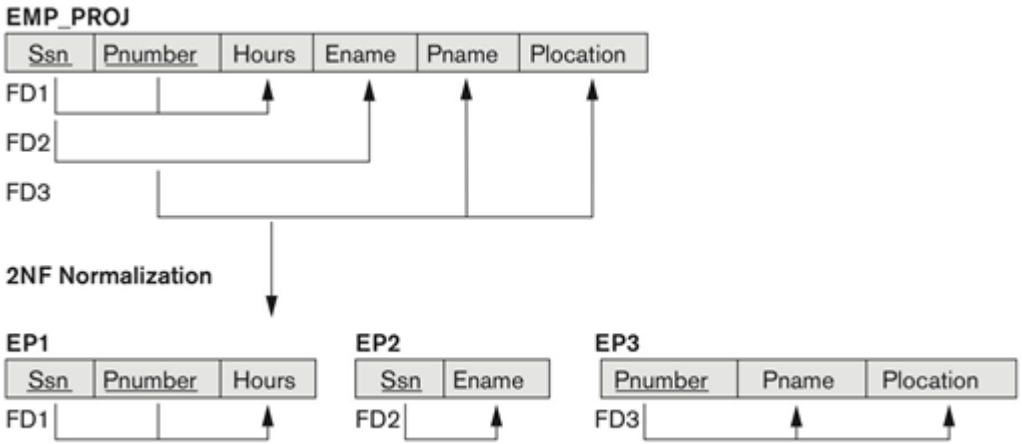
i. it is in 1NF

ii. every non-prime attribute *A* in *R* is fully functionally dependent on the primary key

R can be decomposed into 2NF relations via the process of 2NF normalization or “second normalization”

Figure 14.11 Normalizing into 2NF

(a) Normalizing EMP_PROJ into 2NF relations.



14.3.6 Third Normal Form

- Definition:
  - **Transitive functional dependency:** a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$
- Examples:
  - SSN → DMGRSSN is a **transitive** FD
    - Since SSN → DNUMBER and DNUMBER → DM GRSSN hold
  - SSN → ENAME is **non-transitive**
    - Since there is no set of attributes X where SSN → X and X → ENAME

14.3.6 Third Normal Form

- A relation schema *R* is in **third normal form (3NF)** if

i. it is in 2NF **and**

ii. no non-prime attribute *A* in *R* is transitively dependent on the primary key

R can be decomposed into 3NF relations via the process of 3NF normalization

- NOTE:
  - In $X \rightarrow Y$ and $Y \rightarrow Z$, with *X* as the primary key, we consider this a problem only if *Y* is not a candidate key.
  - When *Y* is a candidate key, there is no problem with the transitive dependency .
  - E.g., Consider EMP (SSN, Emp#, Salary).
    - Here, SSN → Emp# → Salary and Emp# is a candidate key.

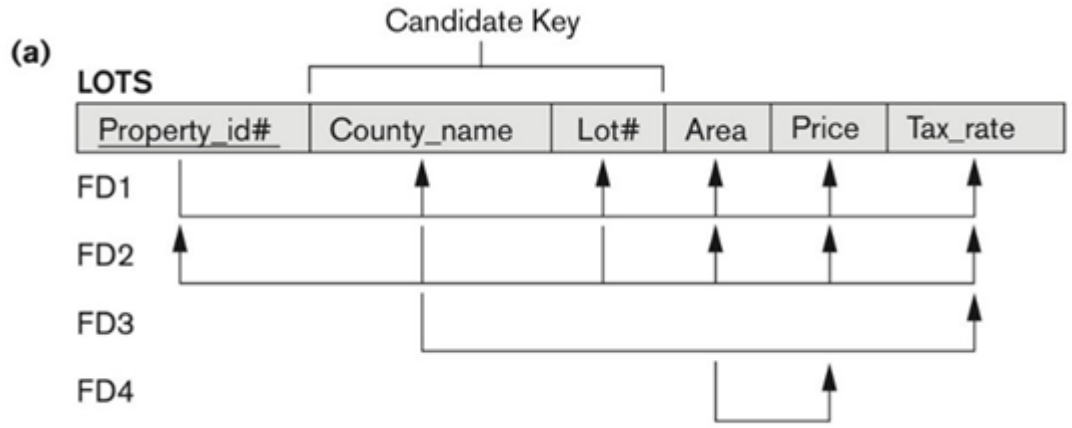
14.4 General Normal Form Definitions (For Multiple Keys)

- 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.**

14.4.1 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*

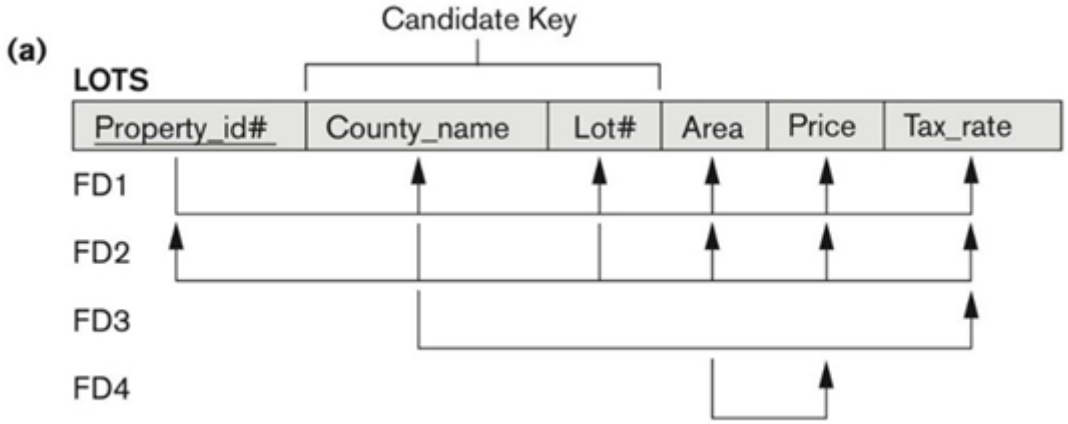
- In Figure 14.12 the F D3
  County_name → Tax_rate violates 2N F.



14.4.1 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*

So second normalization converts LOTS into



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

LOTS2 (County_name, Tax_rate)

14.4.2 General Definition of Third Normal Form

- Definition:
  - **Superkey** of relation schema *R* - a set of attributes S of *R* that contains a key of *R*
  - A relation schema *R* is in **third normal form (3NF)** if whenever a FD $X \rightarrow A$ holds in *R*, then either:

    (a) *X* is a superkey of *R*, or

    (b) *A* is a prime attribute of *R*
- LOTS1 relation violates 3NF because

Area → Price ; and Area is not a superkey in LOTS1. (see Figure 14.12).

14.4.3 Interpreting the General Definition of Third Normal Form (1 of 2)

- Consider the 2 conditions in the Definition of 3NF:
  - A relation schema *R* is in **third normal form (3NF)** if whenever a FD $X \rightarrow A$ holds in *R*, then either:

    (a) *X* is a superkey of *R*, or

    (b) *A* is a prime attribute of *R*
- Condition (a) catches two types of violations :
  - one where a prime attribute functionally determines a non-prime attribute. This catches 2NF violations due to non-full functional dependencies.
  - second, where a non-prime attribute functionally determines a non-prime attribute. This catches 3NF violations due to a transitive dependency.

14.4.3 Interpreting the General Definition of Third Normal Form (2 of 2)

- **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 last slide takes care of the dependencies that **"slip through" (are allowable to) 3NF** but are "caught by" BCNF which we discuss next.

Figure 14.11 Normalizing into 3NF

(b) Normalizing EMP_DEPT into 3NF relations.

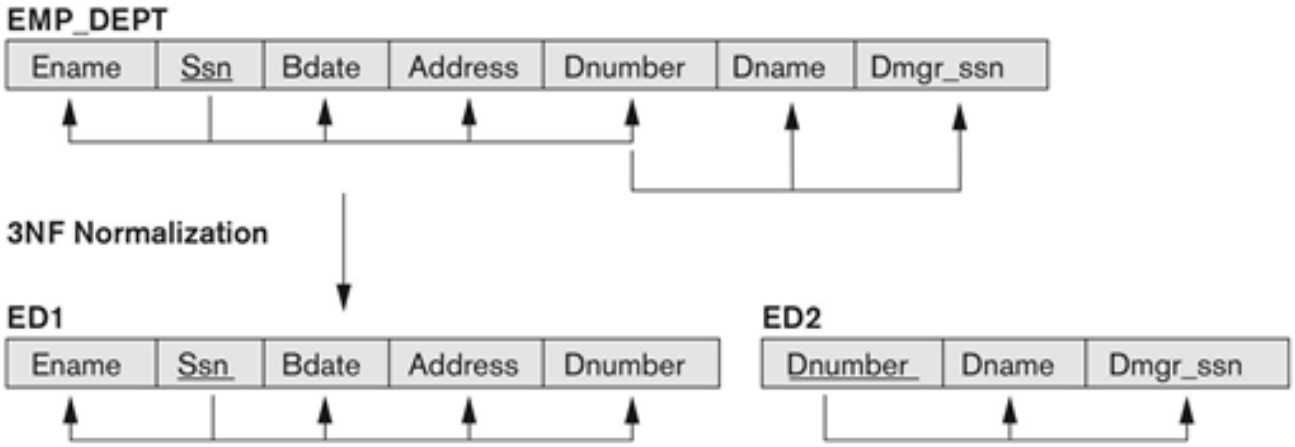
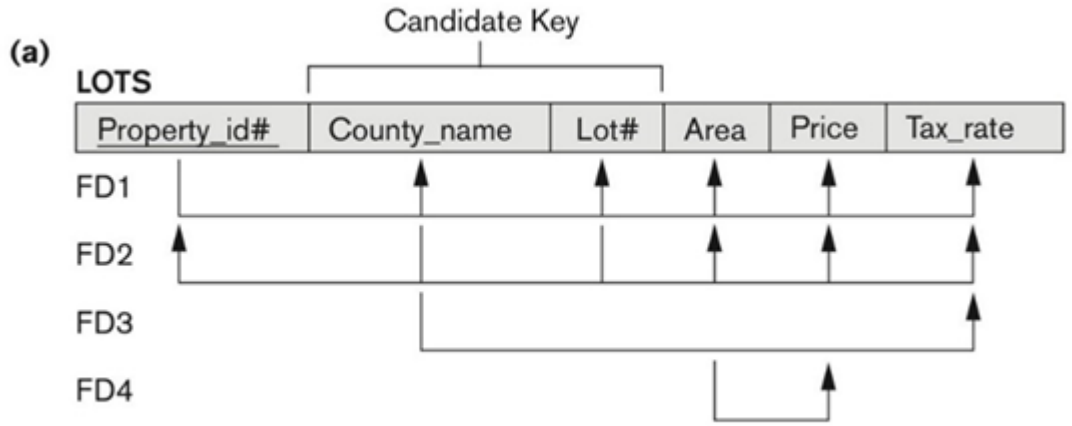
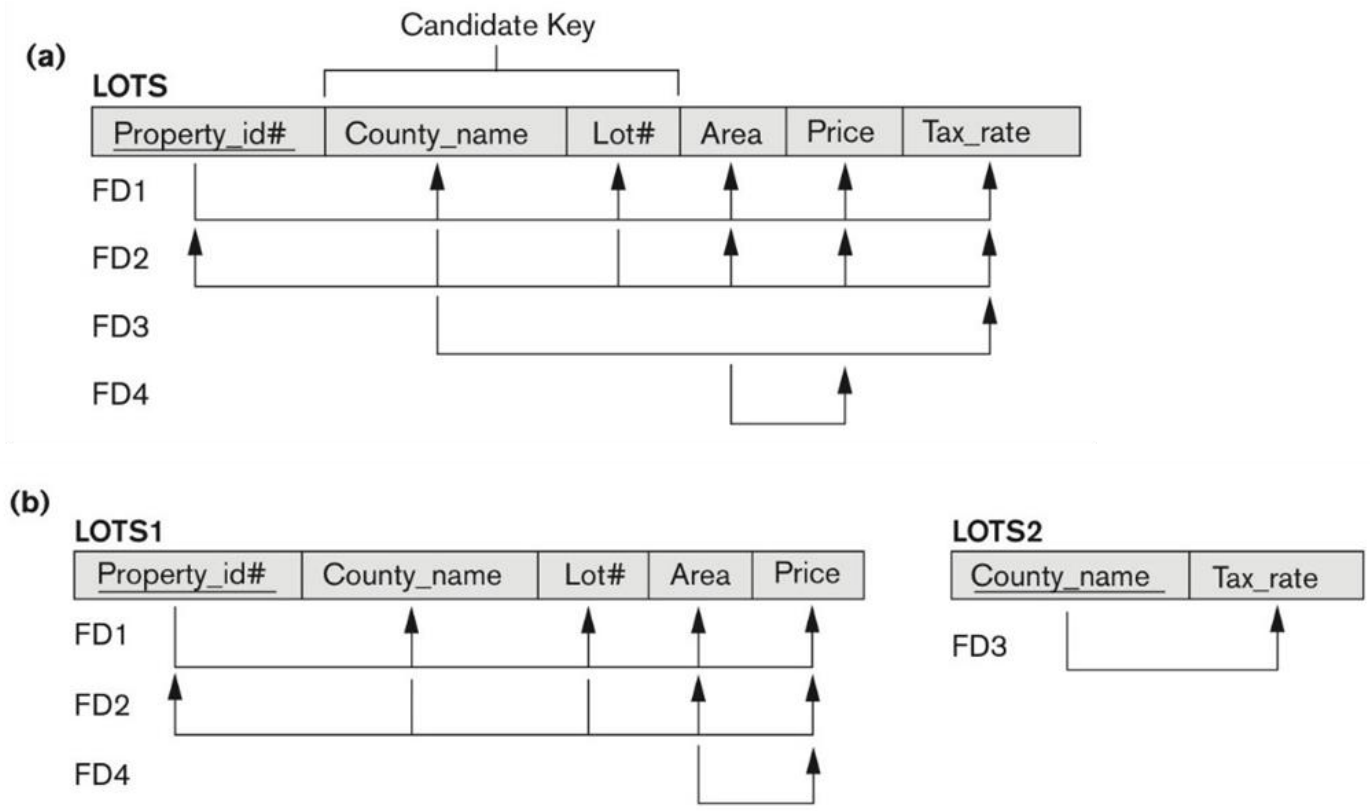


Figure 14.12 Normalization into 2NF and 3NF



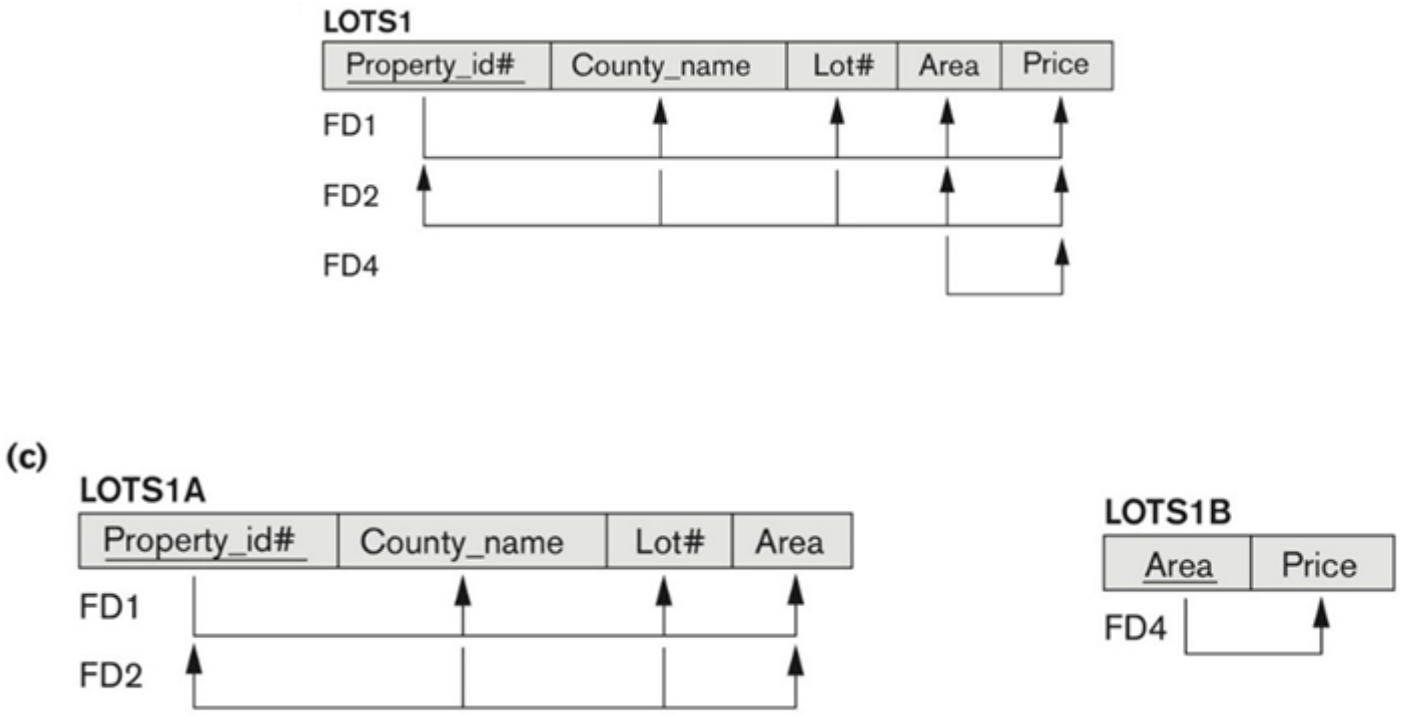
(a) The LOTS relation with its functional dependencies FD1 through FD4.

Figure 14.12 Normalization into 2N F and 3N F



(a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2.

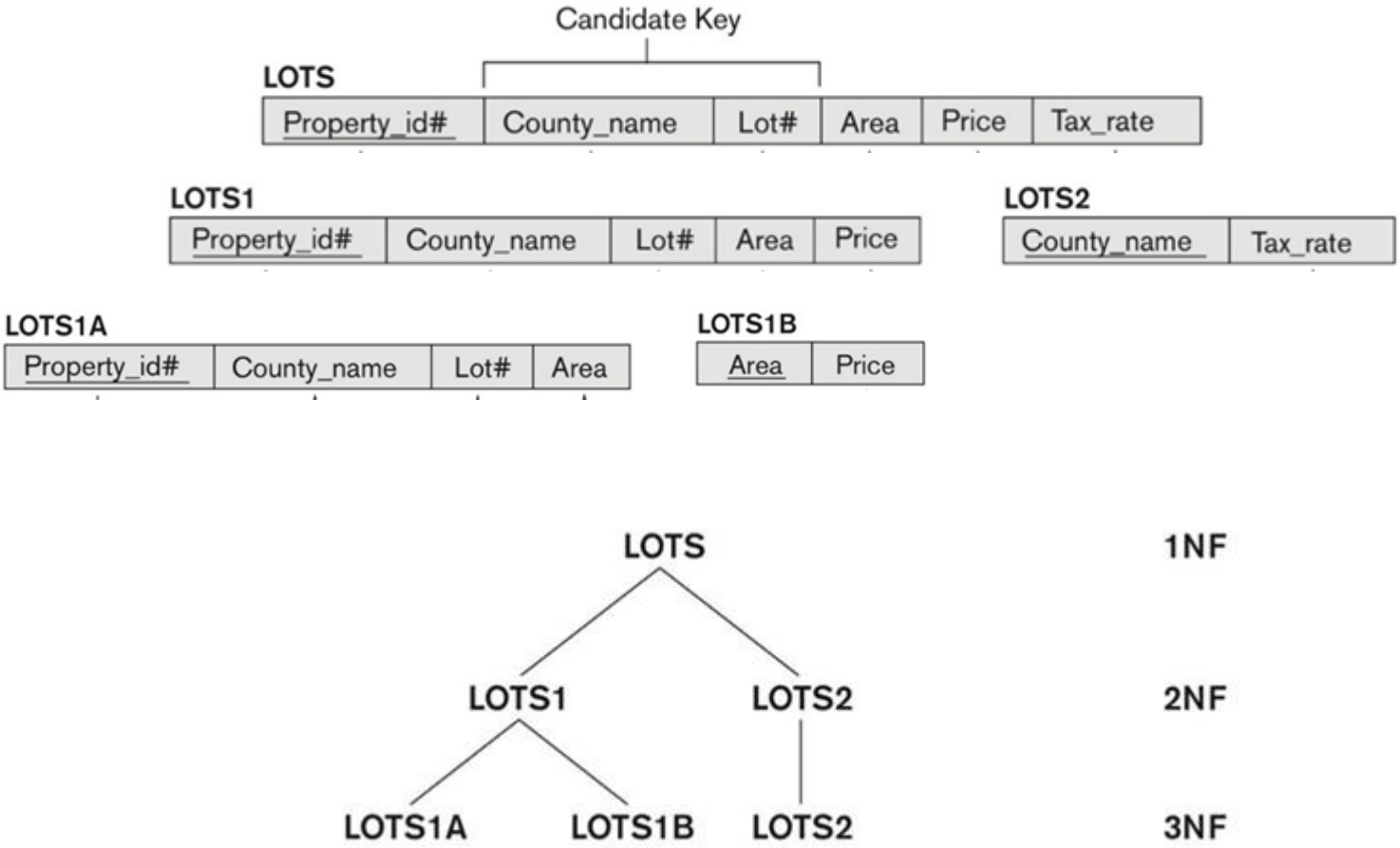
Figure 14.12 Normalization into 2N F and 3N F



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

Figure 14.12 Normalization into 2N F and 3N F

(d) Progressive normalization of LOTS into a 3NF design.

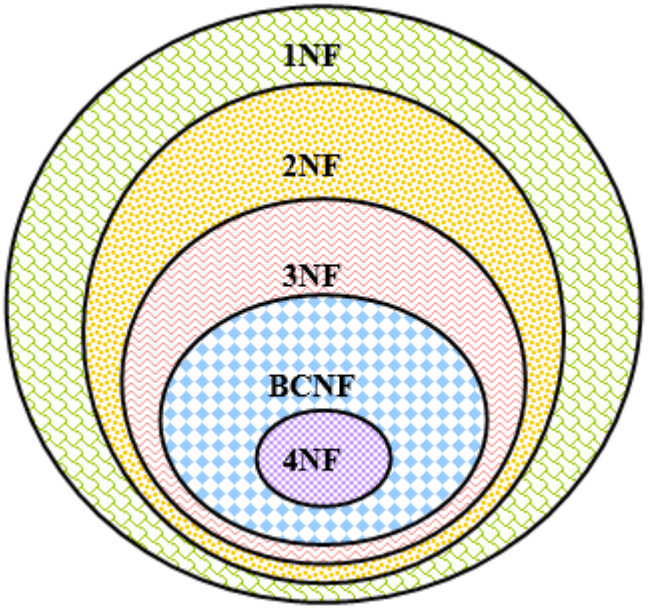


14.5 BCNF (Boyce-Codd Normal Form)

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

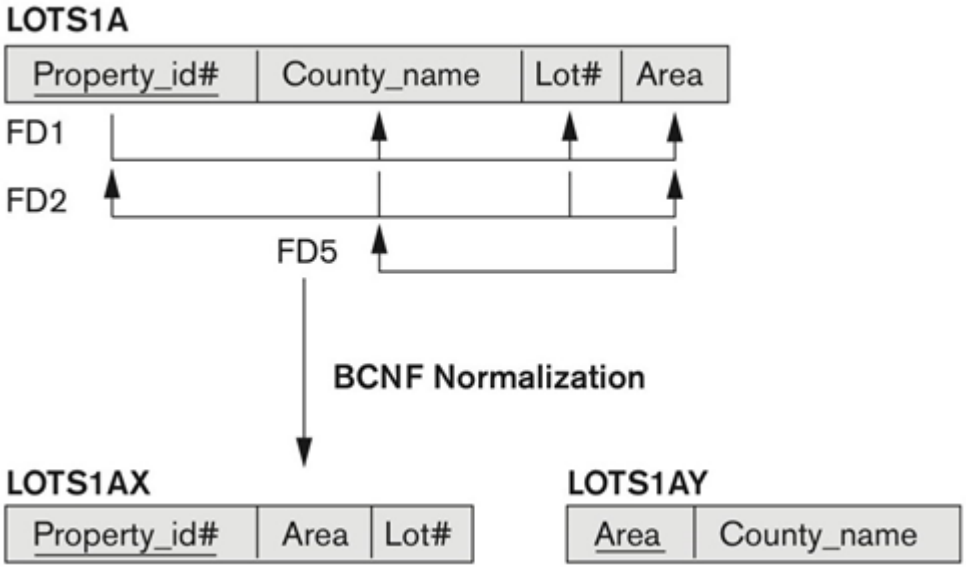
14.5 BCNF (Boyce-Codd Normal Form)

Relationship of Normal Forms



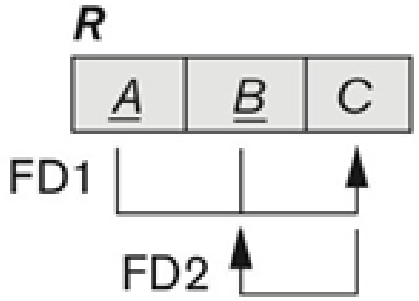
Never leave relations in 1NF or 2NF, most of relations that are in 3NF, are acceptable

Figure 14.13 Boyce-Codd Normal Form



(a) BCNF normalization of LOTS 1A with the functional dependency FD2 being lost in the decomposition.

Figure 14.13 Boyce-Codd Normal Form



(b) A schematic relation with FDs;

it is in 3NF, but not in BCNF due to the f.d. ***C*** → ***B***

Achieving the BCNF by Decomposition

The following are steps to normalize a relation into BCNF:

1. List all of the determinants.

2. See if each determinant can act as a key (candidate keys).

3. For any determinant that is not a candidate key, create a new relation from the functional dependency. Retain the determinant in the original relation.

Achieving the BCNF by Decomposition

The following are steps to normalize a relation into BCNF:

1. List all of the determinants.

2. See if each determinant can act as a key (candidate keys).

3. For any determinant that is not a candidate key, create a new relation from the functional dependency. Retain the determinant in the original relation.

1. student, course
   instructor

2. student, course *YES*
   instructor *NO*

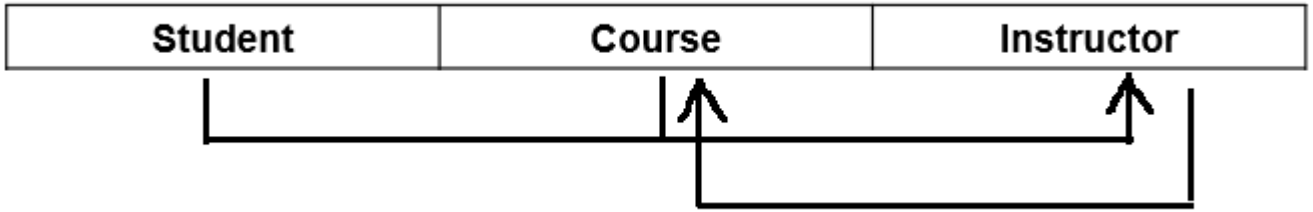
Figure 14.14 A Relation Teach That is in 3NF but Not in BCNF

TEACH

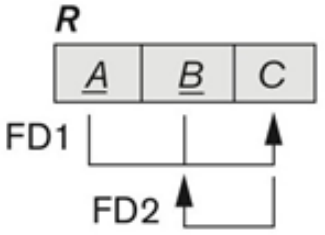
Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

FD1 : {Student, Course → Instructor}
FD2 : Instructor → Course

Achieving the BCNF by Decomposition (1 of 2)

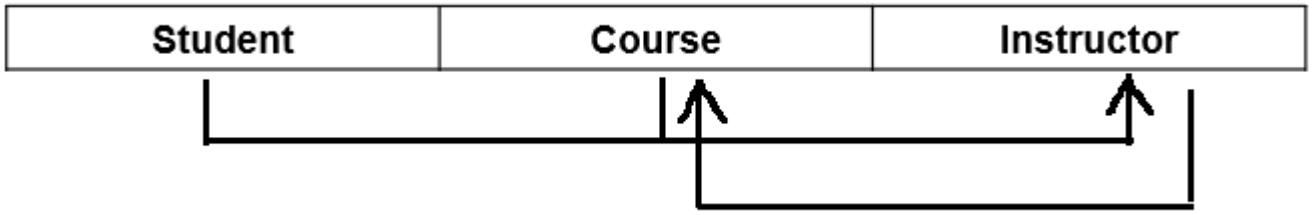


- Two FDs exist in the relation TEACH:
  - FD1: { student, course} → instructor
  - FD2: instructor → course
- {student, course} is a candidate key for this relation and that the dependencies shown follow the pattern
  - So this relation is in 3NF **but not in** BCNF

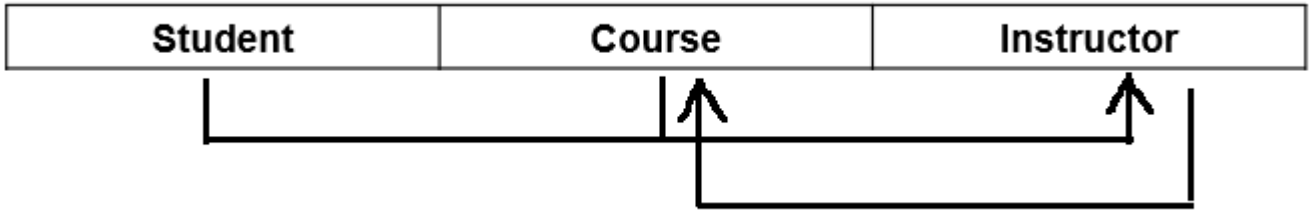


Achieving the BCNF by Decomposition (1 of 2)

- 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.



Achieving the BCNF by Decomposition (2 of 2)



- Three possible decompositions for relation TEACH
  - D1: R1(**student, instructor)** and R2(**student, course**)
  - D2: R1(course, **instructor)** and R2(**course, student)**
  - D3: R1(**instructor**, course) and R2(**instructor, student)**

Achieving the BCNF by Decomposition (2 of 2)

- A test to determine whether a binary decomposition (decomposition into two relations) is non-additive (lossless) is discussed under Property NJB on the next slide. We then show how the third decomposition above meets the property.

Binary Decomposition: Decomposition of a relation *R* into two relations.

Test for Checking Non-Additivity of Binary Relational Decompositions

PROPERTY NJB (non-additive 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 $\left((R_1 \cap R_2) \rightarrow (R_1 - R_2)\right)$ is in F^+, or
- The FD $\left((R_1 \cap R_2) \rightarrow (R_2 - R_1)\right)$ is in F^+.

The notation *F*+ refers to the cover of the set of functional dependencies and includes all FD's implied by *F*.
It is enough to make sure that one of the two FD's actually holds for the nonadditive decomposition into *R*1 and *R*2 to pass this test.

Test for Checking Non-Additivity of Binary Relational Decompositions

- If you apply the NJB test to the 3 decompositions of the TEACH relation:

$$\text{The FD}\left(\left(R_1 \cap R_2\right) \rightarrow \left(R_1 - R_2\right)\right) \text{is in F}^+, \text{or}$$

R1(**student, instructor)** ∩ R2(**student, course**) : student
R1(**student, instructor)** - R2(**student, course**) : course

D1 gives
Student → Instructor or
Student → Course, none of which is true.

Test for Checking Non-Additivity of Binary Relational Decompositions

- If you apply the NJB test to the 3 decompositions of the TEACH relation:

$\text{The FD}\left(\left(R_1 \cap R_2\right) \rightarrow \left(R_1 - R_2\right)\right) \text{is in F}^+, \text{or}$

R1(course, **instructor)** ∩ R2(**course, student): course**

R1(course, **instructor)** - R2(**course, student): student**

D2 gives

Course → Instructor or

Course → Student, none of which is true.

Test for Checking Non-Additivity of Binary Relational Decompositions

- If you apply the NJB test to the 3 decompositions of the TEACH relation:

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

R1(**instructor**, course) ∩ R2(**instructor,** student**) : Instructor**

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.

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

TEACH1

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

TEACH2

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

Chapter 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)