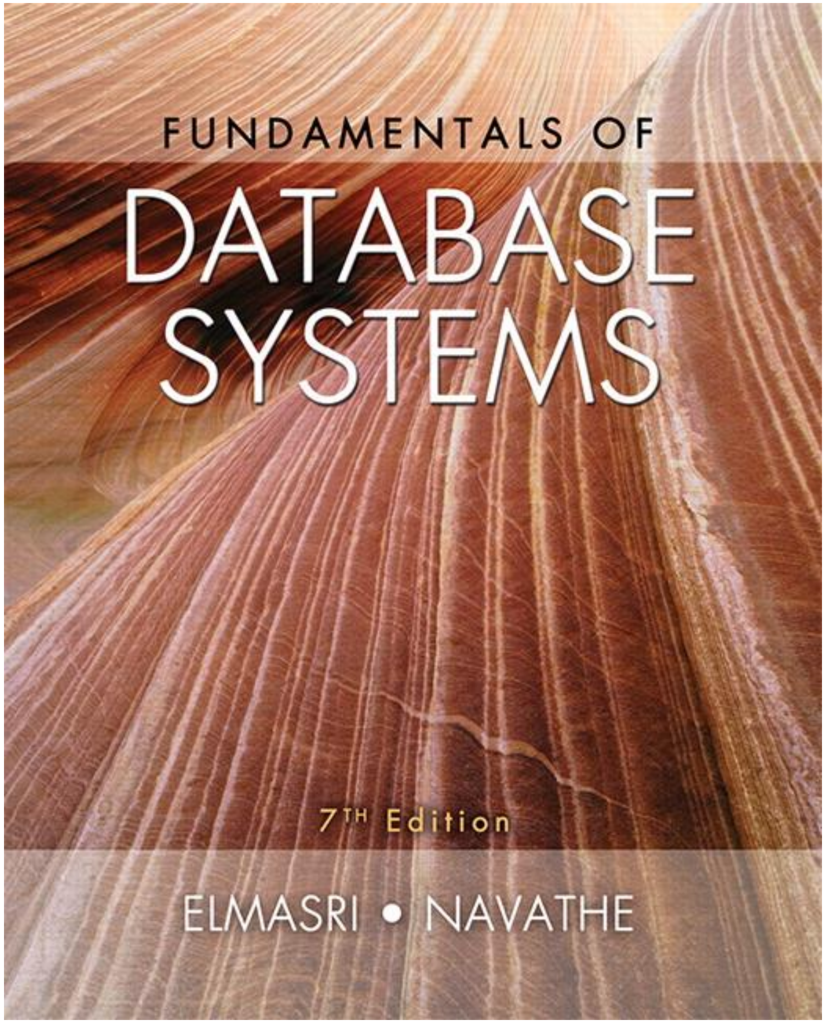
WEEK 11 (Oct 30, 2023)

Basics of Normalization for Relational Databases

(Prof. Shamkant B. Navathe)
sham@cc.gatech.edu


FUNDAMENTALS OF
DATABASE
SYSTEMS
7TH Edition
ELMASRI • NAVATHE

Based on
CHAPTER 14

Basics of Functional Dependencies and Normalization for Relational Databases

Chapter Outline

- 1 Informal Design Guidelines for Relational Databases
  - 1.1 Semantics of the Relation Attributes
  - 1.2 Redundant Information in Tuples and Update Anomalies
  - 1.3 Null Values in Tuples
  - 1.4 Spurious Tuples
- 2 Functional Dependencies (FDs)
  - 2.1 Definition of Functional Dependency

WE COVERED ABOVE MATERIAL IN WEEK 10 (OCT 23, 2023).

Chapter Outline

- 3 Normal Forms Based on Primary Keys
  - 3.1 Normalization of Relations
  - 3.2 Practical Use of Normal Forms
  - 3.3 Definitions of Keys and Attributes Participating in Keys
  - 3.4 First Normal Form
  - 3.5 Second Normal Form
  - 3.6 Third Normal Form
- 4 General Normal Form Definitions for 2NF and 3NF (For Multiple Candidate Keys)
- 5 BCNF (Boyce-Codd Normal Form)

ALL OF ABOVE TO BE COVERED IN WEEK 11 (TODAY)

Chapter Outline

- 6 Multivalued Dependency and Fourth Normal Form
- 7 Join Dependencies and Fifth Normal Form

ABOVE SECTIONS WILL NOT BE COVERED AT ALL

REVIEW

A review of what we learnt
and what you will do
in the tutorial discussion
this week

1. Informal Design Guidelines for Relational Databases – Guideline 1

- GUIDELINE 1: Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).
- Bottom Line: *Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.*

Informal Design Guidelines for Relational Databases – Guideline 2

- GUIDELINE 2:
  - Design a schema that does not suffer from update anomalies
    - Insertion anomalies
    - Deletion anomalies
    - Modification anomalies
  - If there are any anomalies present, then note them so that applications can be made to take them into account
  - *These are typically introduced for reasons such as attributes are needed for reporting (e.g.,besides dept_no., department-name, manager-name is required) or accounting purposes (e.g., besides invoice_no., invoice_amount is required)*
  - *This is referred to as De-Normalization*

Informal Design Guidelines for Relational Databases – Guideline 3

- 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

Informal Design Guidelines for Relational Databases – Guideline 4

GUIDELINE 4:

- Avoid generation of "spurious data" when tables are joined – an absolute "MUST".
- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- Generating bad data cannot be accepted at any cost.
- The "lossless join" property is used to guarantee that join operation will not create bad data.
  - 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.

Properties of Decompositions

- 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.

If the losslessness (non-additivity) of joins is not guaranteed, there will be chaos in terms of lot of spurious data generated by database queries and transactions

2. Functional Dependencies – as a tool to analyze designs

- 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

2. Functional Dependencies – further concepts

- Inference Rules
  - We introduced Armstrong's 3 basic rules for inferring new FDs from a given set of FDs
- Closures
  - We defined Closure of FDs as the set of all FDs that can be derived from a given set.
  - We defined the Closure of an attribute as the set of all attributes that are functionally dependent on a given attribute.

2. Functional Dependencies – further concepts – contd.

- Cover
  - We defined that a set of FDs X covers another set Y if all FDs in set Y can be inferred from set X
- Equivalence
  - We defined the concept of equivalence among two sets of FDs F and G based on F covering G and G covering F
- Minimum Cover
  - We defined how to compute the minimum cover F_{min} of a set F as an equivalent set where there are no extraneous attributes on the LHS of any FD in F_{min} and there is no redundant FD in $F_{min.}$

3 Normal Forms Based on Primary Keys

Now we turn our attention to Section 3 of this Chapter.

- 3.1 Normalization of Relations
- 3.2 Practical Use of Normal Forms
- 3.3 Definitions of Keys and Attributes Participating in Keys
- 3.4 First Normal Form
- 3.5 Second Normal Form
- 3.6 Third Normal Form

3.1 Normalization of Relations (1)

- **Normalization:**
  - The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations
  - (may be considered as “improvement”, “purification” of a given design to make it better so that anomalies and redundancy are eliminated)
- **Normal form:**
  - Condition using keys and FDs of a relation to certify whether a relation schema is in a particular state of “goodness”.

Normalization of Relations (2)

- 2NF, 3NF, BCNF
  - based on keys and FDs of a relation schema
- 4NF
  - based on keys, multi-valued dependencies : MVDs;
- 5NF
  - based on keys, join dependencies : JDs
- Additional properties/conditions necessary to ensure a good relational design are: lossless join, and dependency preservation.

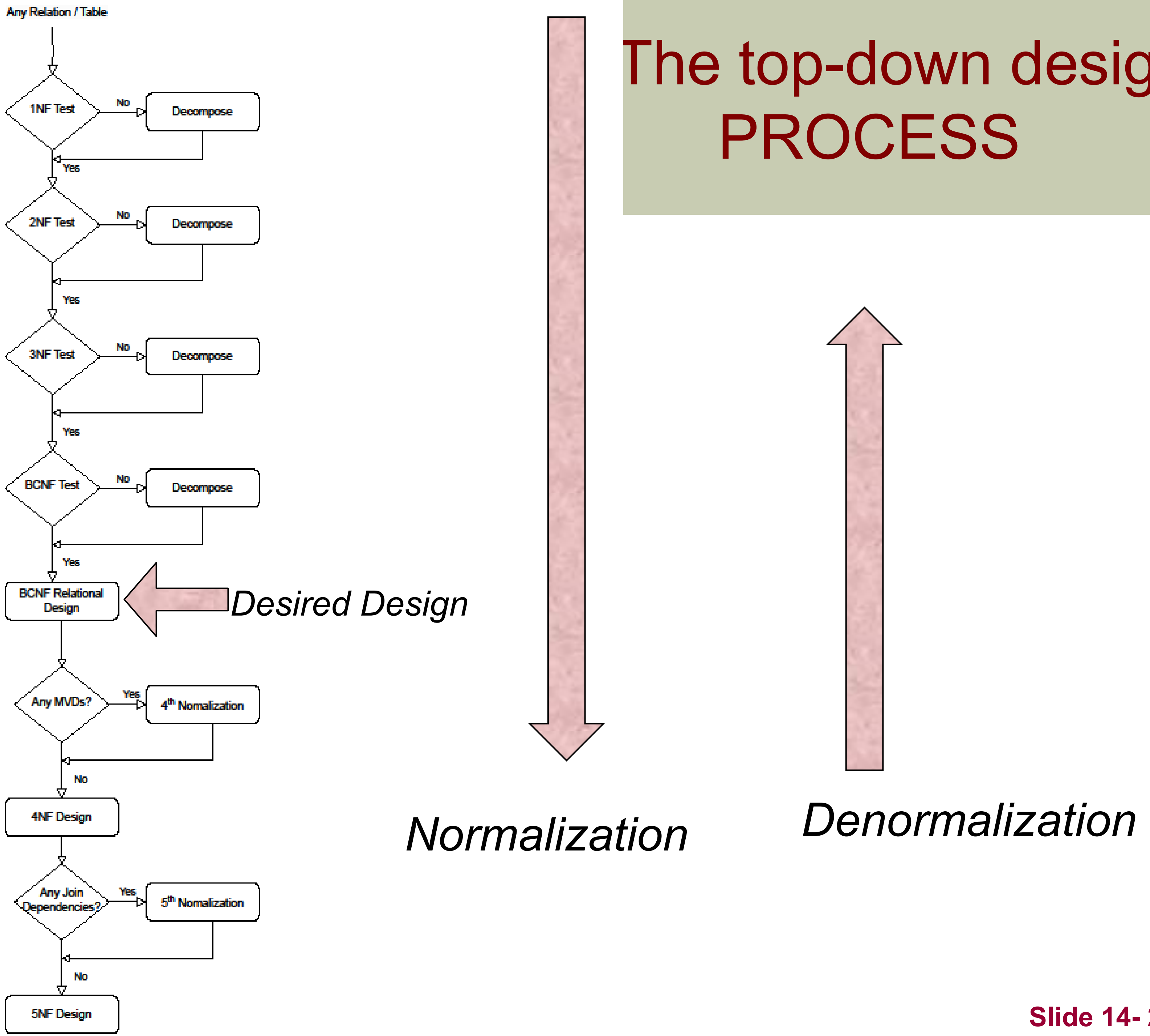
We will consider relational design upto BCNF only because for all practical purposes, that is a standard practice in industry and higher normal forms are not relevant in common commercial database designs.

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 we normalize up to 3NF and BCNF.
  - 4NF, 5NF 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

The top-down design PROCESS



3.3 Definitions of Keys and Attributes Participating in Keys (1)

- 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 (2)

- If a relation schema has more than one key, each is called a **candidate** key.
  - One of the candidate keys is *arbitrarily* designated to be the **primary key**, and the others are called candidate keys (alternate keys).
- A **Prime attribute** must be a member of *some* candidate key
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

Definitions of Keys and Attributes Participating in Keys (3)

Person (Ssn, Email, Name, Birthdate, Income, Postal_code)

- (Ssn, Name) : superkey
- (Name, Birthdate): not guaranteed to be a key
- Ssn: a minimal superkey : a candidate key
- Email: key – a candidate key, assuming every person is required to have a unique email
- (Ssn, Age, Income) – superkey? - YES
- (Name, Birthdate, Postal_code) : is used many times by police etc. to identify a unique individual – behaves like a candidate key but may not be a declared candidate key.
- In SQL data definition UNIQUE specification can be used to declare candidate keys. E.G.,

PRIMARY KEY (Ssn)

UNIQUE (Email)

Definitions of Keys and Attributes Participating in Keys (4)

- Example 2:

R (K1, K2, C1, C2, D, E)

If (K1, K2) → C1, C2, D, E then (K1, K2) is a candidate key.

If (C1, C2) → K1, K2 then (C1,C2) is also a candidate key.

Is (K1, K2 , D) a superkey?

Is (K1, C1) a superkey?

Is (C1, C2 , E) a superkey?

Is (C1, C2 , K1, K2) a superkey?

Is (K1, K2 , D, E) a superkey?

If we were to store this relation R, we would typically designate the shorter of the (K1, K2) and (C1, C2) as "the primary key".

Various definitions of Keys

SUPERKEYS

↓

MINIMAL SUPERKEYS: CANDIDATE KEYS

↓

DESIGNATED CANDIDATE KEY : PRIMARY KEY

*In Indexing literature we use the term "**secondary keys**" to refer to any attributes that may be used to define an index. For example, in the relation: PERSON (Ssn, name, salary, age, postal_code) We may use "age" and "postal_code" as Indexing keys, or secondary keys for efficient access for queries that are based on age or postal_code..*

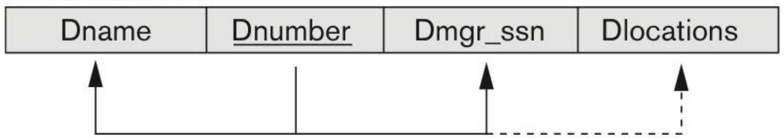
3.4 First Normal Form

- 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)

DEPARTMENT



(b)

DEPARTMENT

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

(c)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	<u>Dlocation</u>
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.

First Normal Form – functional dependency analysis (1)

Functional Dependency argument:

- DEPARTMENT (<u>Dnumber</u>, Dname, Dmgr_ssn, Dlocations)

Dnumber is primary key.

For a relation to be in 1NF every attribute must functionally be dependent on the primary key.

$\mathcal{F}$: {Dnumber $\rightarrow$ Dname, Dmgr_ssn}

But Dnumber $\not\rightarrow$ Dlocations. (This FD does <u>not</u> exist).

Hence the DEPARTMENT relation does not meet the 1NF requirement.

REMEDY: Decomposition into:

DEPARTMENT1 (<u>Dnumber</u>, Dname, Dmgr_ssn)

DEPARTMENT2 (<u>Dnumber</u>, <u>Dlocation</u>)

It preserves original FDs in F above.

DEPARTMENT2 has NO FD.

Figure 14.10 Normalizing nested relations into 1NF

(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 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.

First Normal Form –functional dependency analysis (2)

Functional Dependency argument:

- EMP_PROJ (<u>Ssn</u>, Ename, PROJ (Pno, Hours))

Ssn is the primary key.

Here, the nested relation PROJ behaves like a composite nested attribute with its local key being Pno, analogous to a weak entity type PROJ under a strong entity type EMP.

Hence Ssn $\not\rightarrow$ PROJ. Only FD is Ssn $\rightarrow$ Ename. Hence, the relation is NOT in 1NF.

And (Ssn, Pno) $\rightarrow$ Hours

We remedy the situation by creating a new relation that accommodates the nested relation with a joint primary key:

EMP_PROJ1 (<u>Ssn</u>, Ename)

EMP_PROJ2 (<u>Ssn</u>, <u>Pno</u>, Hours)

3.5 Second Normal Form (1)

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

Second Normal Form (2)

- A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on the primary key

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

Figure 14.11 Normalizing into 2NF (and 3NF)

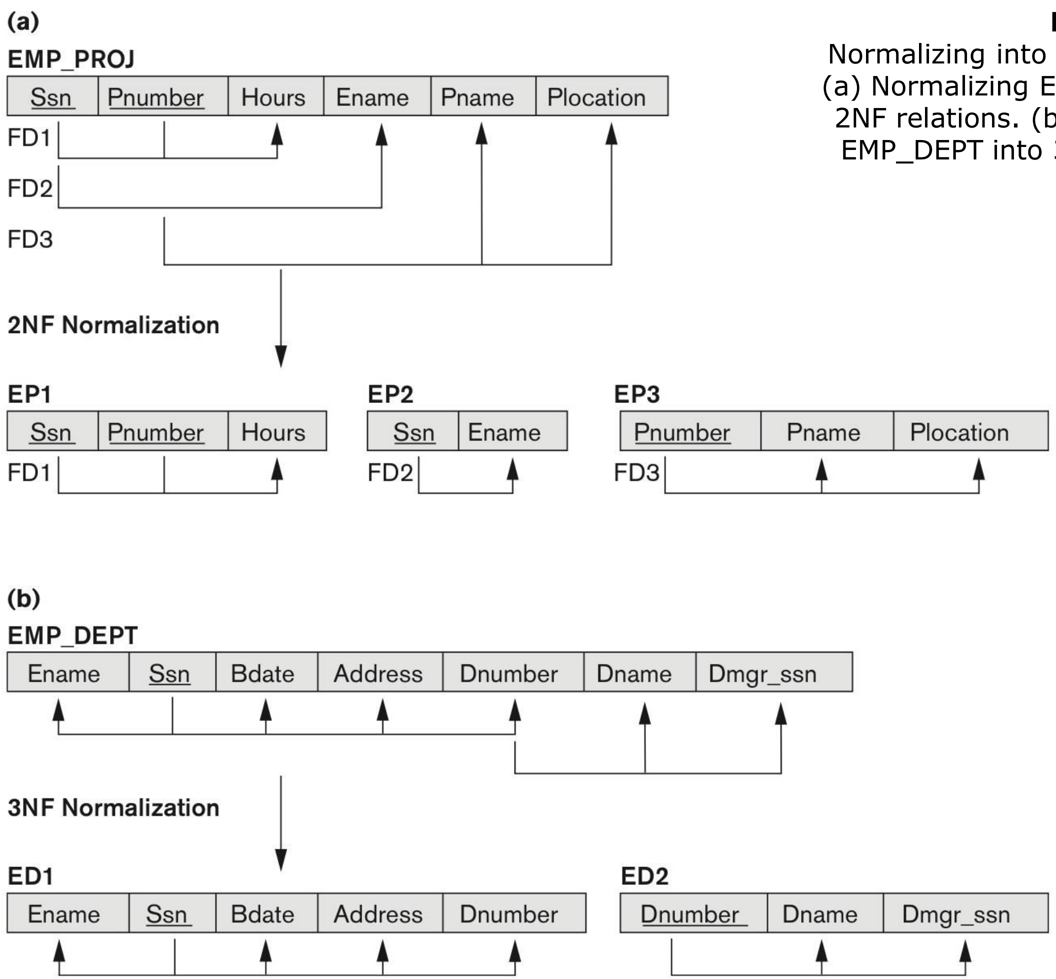


Figure 14.11
Normalizing into 2NF and 3NF. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

Second Normal Form –functional dependency analysis

- Functional Dependency argument:

EMP_PROJ(Ssn, Pnumber, Hours, Ename, Pname, Plocation)

Violation of 2NF: non-full-functional dependency of attributes Ename and Pname on the primary key.

The only attribute fully functionally dependent on the primary key is Hours: Ssn, Pnumber → Hours. (FD1)

But, Ssn → Ename. (FD2) and

Pnumber → Pname , Plocation. (FD3).

We say that FD2 and FD3 are the cause of 2NF violation.

REMEDY: Decompose so as to achieve full functional dependence on primary key in each relation:

EP1 (Ssn, Pnumber ,Hours); EP2 (Ssn,Ename); and EP3 (Pnumber, Pname , Plocation).

All FDs (FD1, FD2, FD3) are preserved.

(Recollect that all update anomalies are eliminated)

Second Normal Form –additional comments

Consider a relation

- R (<u>K1, K2, K3</u>, A, B, C, D)

Assume the FDs are $\mathcal{F}$: {K1 → A; K2 → B; K3 → C ;

(K1, K2, K3) → A,B,C,D}.

What NF?: Only 1NF. Because of the first 3 FDs.

Decomposition?

R1 (<u>K1</u> , A)

R2 (<u>K2</u> , B)

R3 (<u>K3</u> , C)

R4 (<u>K1, K2, K3</u>, D)

Second Normal Form –additional comments

Now, consider a relation

- S (<u>K1, K2, K3</u>, A, B, C)

Assume the FDs are: $\mathcal{F}$: {K1 → A; K2 → B; K3 → C;

(K1, K2, K3) → A,B,C }.

There is no attribute that is fully functionally dependent on the entire primary key.

What NF?: Only 1NF. Because of the 3 FDs.

Decomposition?

R1(K1, A)

R2(K2, B)

R3(K3, C)

R4 (<u>K1, K2, K3</u>)

Are we done?

Second Normal Form –additional comments

Consider another relation

- S (<u>K1, K2, K3</u>, A, B, C, D)

Assume the FDs are: $\mathscr{F}$: { (K1, K2) → A; (K2, K3) → C; (K1, K2, K3) → A,B,C,D }.

Here again, there is partial dependence of A and C on the entire key: (K1, K2, K3) .

What NF?: Only 1NF. Because of this partial dependence.

Decomposition?

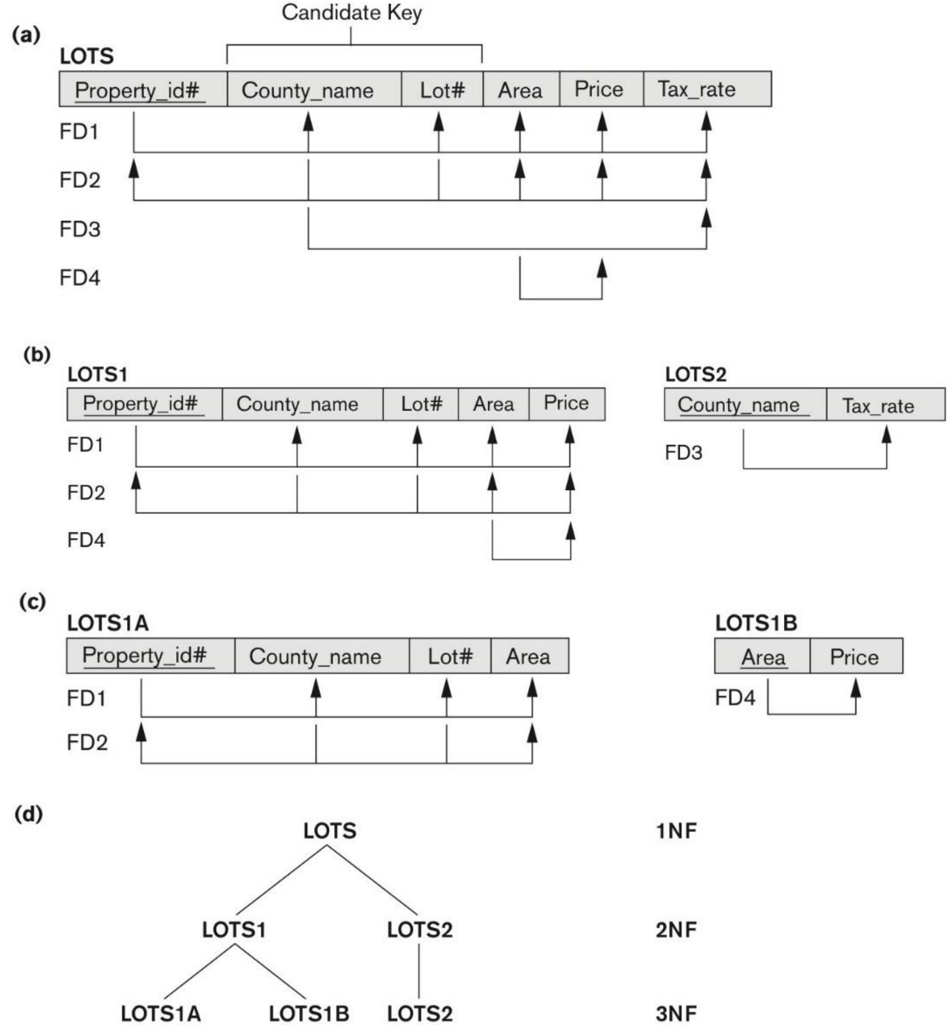
R1(<u>K1, K2</u>, A)

R2(<u>K2, K3</u>, C)

R3(<u>K1, K2 K3</u> , B, D)

Figure 14.12 Another Example: Normalization into 2NF (and 3NF)

Figure 14.12
Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Progressive normalization of LOTS into a 3NF design.



3.6 Third Normal Form (1)

- Consider the relation in Fig. 14.11:

EMP_DEPT (Ename, <u>Ssn</u>, Bdate, Address, Dnumber, Dname, Dmgr_ssn)

Definition:

- **Transitive functional dependency:** a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$

- In EMP_DEPT :
  - 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 . It is directly and not transitively dependent on SSN.

Third Normal Form (2)

- A relation schema R is in **third normal form (3NF)** if it is in 2NF *and* no non-prime attribute A in R is transitively dependent on the primary key.

(*The above is a primary-key based and successive decomposition based criterion for 3NF*).

- R can be decomposed into 3NF relations via the process of 3NF normalization that removes the transitive dependency (in the relation that already satisfies 2NF).
- NOTE:
  - In X → Y and Y → Z, with X as the primary key, we consider this a problematic case only if Y is not a candidate key.
  - When Y is a candidate key, there is no problem with the transitive dependency, and it does not cause 3NF violation.
  - E.g., Consider EMP (<u>SSN</u>, Emp#, Salary).
    - Here, SSN → Emp# → Salary; and Emp# → SSN. i.e., it is a candidate key.
    - Hence, there is no problematic transitive dependency; So EMP is in 3NF and Salary is an attribute that is directly dependent on the key.

Third Normal Form – functional dependency analysis

EMP_DEPT (Ename, <u>Ssn</u>, Bdate, Address, Dnumber, Dname, Dmgr_ssn).

Here, <u>Ssn</u> → EMP_DEPT ; Ssn is the primary key.

However, Dnumber → (Dname,Dmgr_ssn) is an FD that is among non-prime attributes and it causes the transitive dependency (of Dname and Dmgr_ssn on Ssn via Dnumber),

Transitive Dependency : Ssn → Dnumber → (Dname,Dmgr_ssn).

Because of the above transitive dependency the above relation is not in 3NF

Is it in 2NF?

Why?

The third normalization should remove the transitive dependency:

Hence third normalization produces (see Fig. 14.11):

ED1 (Ename, <u>Ssn</u>, Bdate, Address, Dnumber)

ED2 (<u>Dnumber</u>, Dname, Dmgr_ssn).

Note: It preserves the original FDs due to the foreign key attribute Dnumber in ED1. By applying the transitive dependency Armstrong’s rule, we can get it back.

Successive Normalization (1)

- Going through the process of top-down design followed in most organizations, a given relation schema can be refined successively to achieve the higher normal forms. (see the successive analysis flowchart we saw earlier)
- This is illustrated in the LOTS example in Fig. 14.12

LOTS (Propertyid#, County_name, Lot#, Area, Price, Tax_rate),

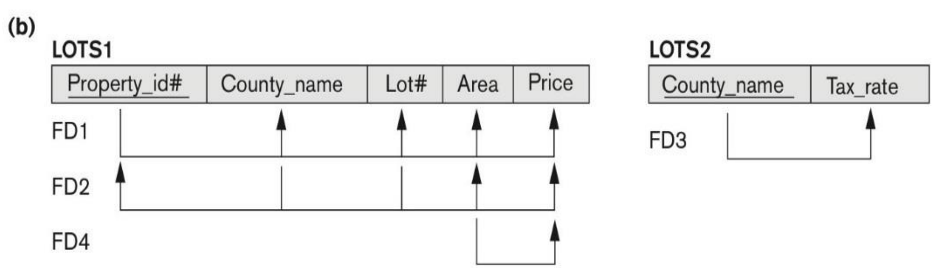
FD1: Propertyid# → LOTS; primary key.

FD2 : (County_name, Lot#) → LOTS ; candidate key

FD3: County_name → Tax_rate

FD4: Area → Price

2nd Normalization: Remove the non-full-functional dependence of Tax_rate on County_name which is a part of the Candidate Key. Gives:



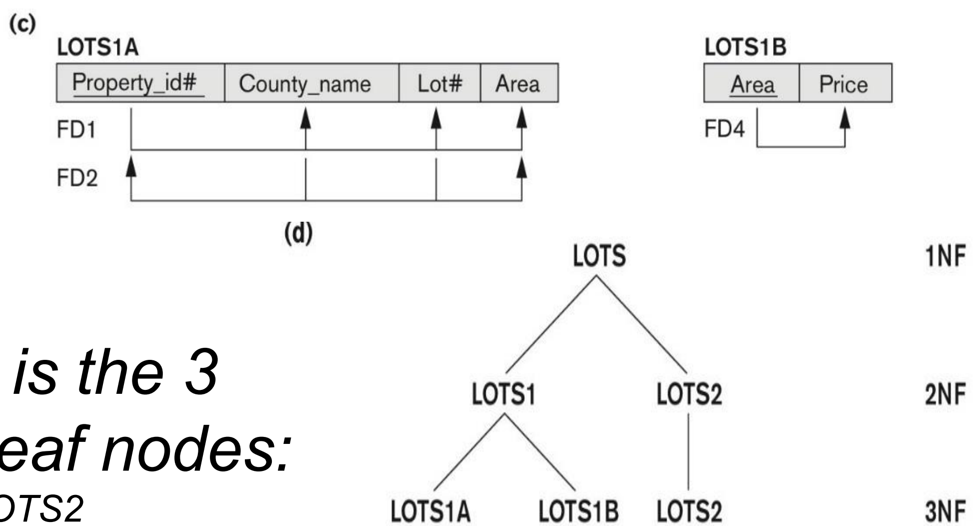
Successive Normalization (2)

FD4: Area → Price shows that

Propertyid# → Area → Price is a case of transitive dependence of Price on the Propertyid# via the non-prime attribute Area.

This is a cause for 3 NF violation.

Hence, LOTS 1 is not in 3NF and must be decomposed into LOTS1A and LOTS1B:



Final design is the 3 relations at leaf nodes:

LOTS1A, LOTS1B,LOTS2

All FDs have been preserved.

Normal Forms Defined Informally

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

4. General Normal Form Definitions (For Multiple Keys) (1)

- The above definitions so far considered 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.*

4.1 General Definition of 2NF (For Multiple Candidate Keys)

Definition: 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 FD

  County_name → Tax_rate violates 2NF.

County_name is a prime attribute.

So second normalization converts LOTS into

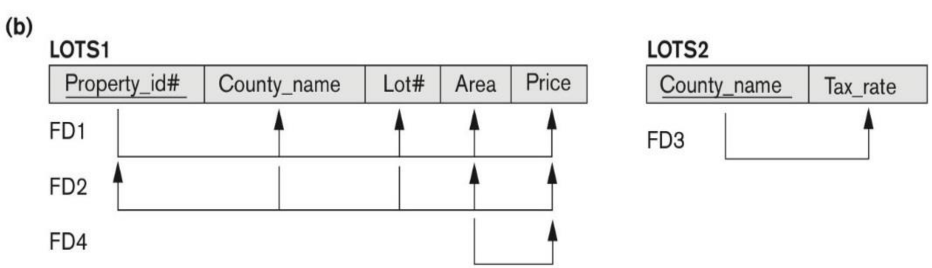
LOTS1 (<u>Property_id#,</u> County_name, Lot#, Area, Price)

LOTS2 (<u>County_name</u>, Tax_rate)

4.2 General Definition of Third Normal Form

- **Superkey** of relation schema R - a set of attributes S of R that contains a key of R
- ***Definition:*** A relation schema R is in **third normal form (3NF)** if whenever a FD 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.



4.3 Interpreting the General Definition of Third Normal Form

- Consider the 2 conditions in the Definition of 3NF:

  A relation schema R is in **third normal form (3NF)** if whenever a FD X → 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 (or a set of them) functionally determines another non-prime attribute (or a set). This catches 3NF violations due to a transitive dependency

4.3 Interpreting the General Definition of Third Normal Form (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.

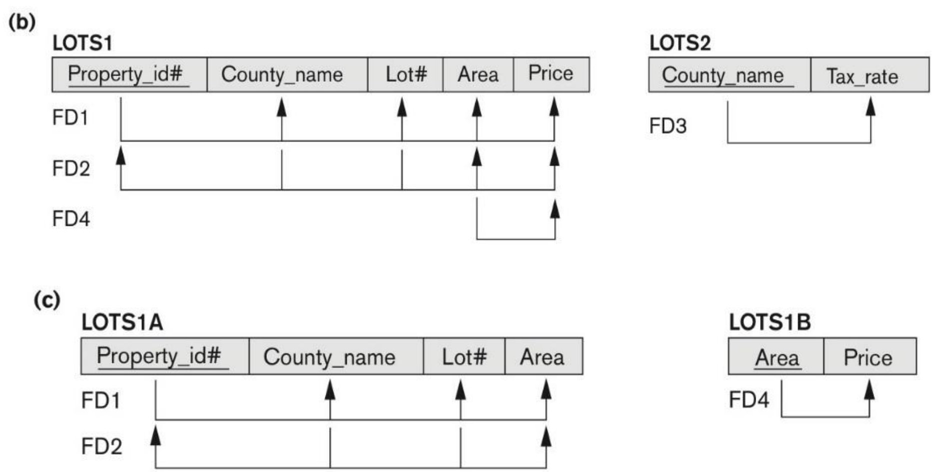
- The condition (b) from the last slide takes care of the dependencies that "slip through" (are allowable to) 3NF. Such dependencies are "caught by" BCNF which we discuss next.

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, or 3NF+.
- The general acceptable goal in practice is to have each relation in BCNF (or at least 3NF)

5. BCNF (Boyce-Codd Normal Form)

- We already have the 3 relations from LOTS which are in 3NF as shown earlier:



Our final 3NF design consisted of LOTS1A, LOTS1B and LOTS2.
Now, suppose Area → County_name *in LOTS1A relation. E.g., If area is 1000 m^2 or 2000 m^2 then it is county Fulton, if it is 3000 m^2 then county is Cobb, etc.*
So we call this a new FD5: Area → County_name

Figure 14.13 Boyce-Codd normal form

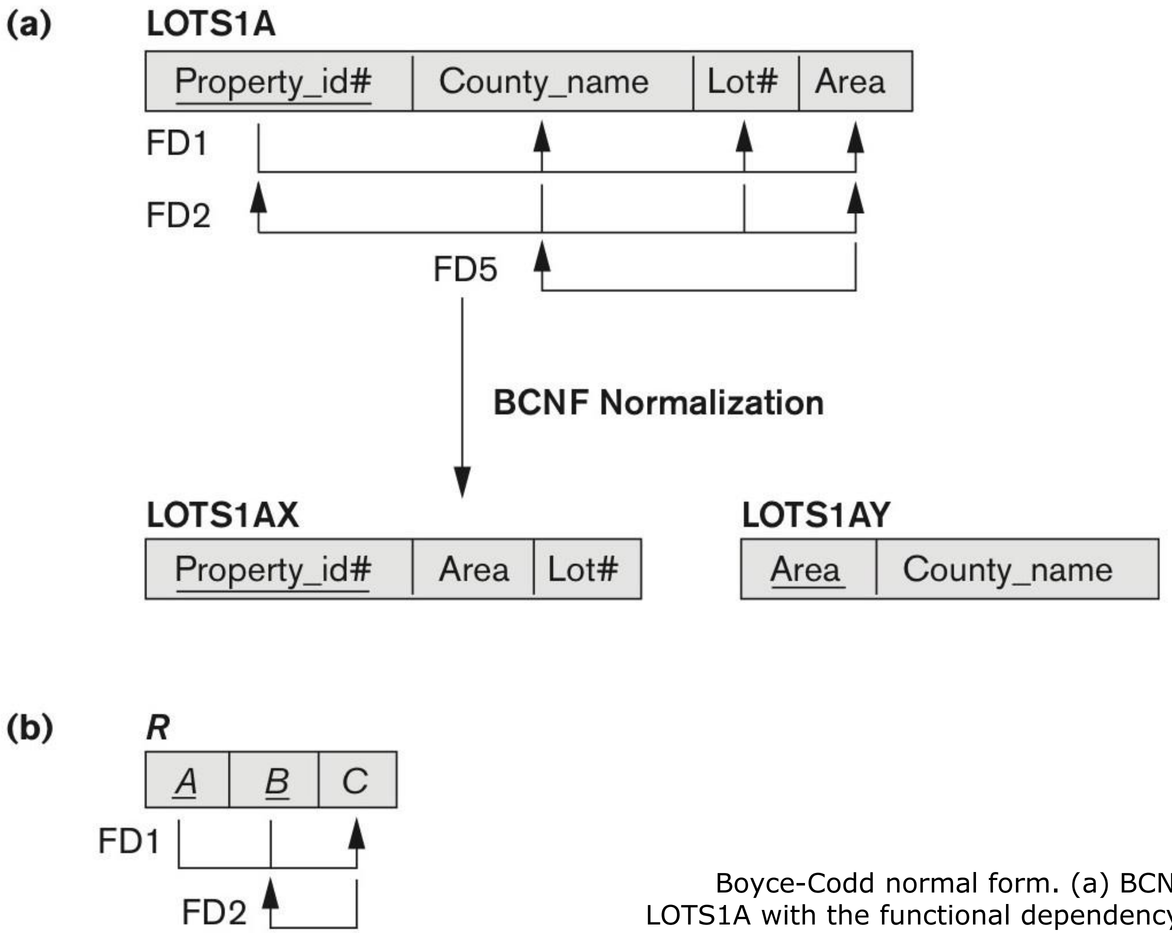


Figure 14.13
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF due to the f.d. C → B.

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

Figure 14.14
A relation TEACH that is in 3NF but not BCNF.

Achieving the BCNF by Decomposition (1)

- Two FDs exist in the relation TEACH:
  - fd1: { student, course} -> instructor
  - fd2: instructor -> course
- {student, course} is a candidate key for this relation and that the dependencies shown follow the pattern in Figure 14.13 (b).
  - So this relation is in 3NF *but not in* BCNF
- A relation **NOT** in BCNF should be decomposed so as to achieve BCNF and meet the property that every FD has its LHS which must be a superkey.
- However, this decomposition possibly forgoes the preservation of all functional dependencies in the decomposed relations.

Achieving the BCNF by Decomposition (2)

- 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 losslessness (non-additivity) property after decomposition.
- Out of the above three, only the 3rd decomposition D3 will not generate spurious tuples after join.(and hence has the non-additivity property).

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

Test for checking non-additivity of Binary Relational Decompositions

- **Testing Binary Decompositions for Lossless Join (Non-additive-Join–Binary: NJB) 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 f.d. $((R1 \cap R2) \rightarrow (R1 - R2))$ is in F^+, or
    - The f.d. $((R1 \cap R2) \rightarrow (R2 - R1))$ is in 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.

Question: What about n-ary decompositions?

Answer: Alg. 15.3 in Chapter 15.

General Procedure for achieving BCNF when a relation fails BCNF

Here we make use of the algorithm from Chapter 15 (Algorithm 15.5):

- Let R be the relation not in BCNF, let X be a subset-of R, and let $X \rightarrow Y$ be the FD that causes a violation of BCNF. Then R may be decomposed into two relations:
- (i) $R - Y$ and (ii) $X \cup Y$.
- If either $R - Y$ or $X \cup Y$. is not in BCNF, repeat the process.

Note that the f.d. that violated BCNF in TEACH was Instructor $\rightarrow$Course. Hence its BCNF decomposition would be :

(TEACH – COURSE) and (Instructor U Course), which gives

the relations: (Instructor, Student) and (Instructor, Course) that we obtained before in decomposition D3.

Summary of What we learnt

- Informal Design Guidelines for Relational Databases
- Functional Dependencies (FDs) and how to infer new FDs; properties related to 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)
- How to decompose a relation that may be in 3NF and not BCNF.