Normalization

5.3

normalization is

a technique of organizing the data into multiple related tables, to minimize DATA REDUNDANCY.

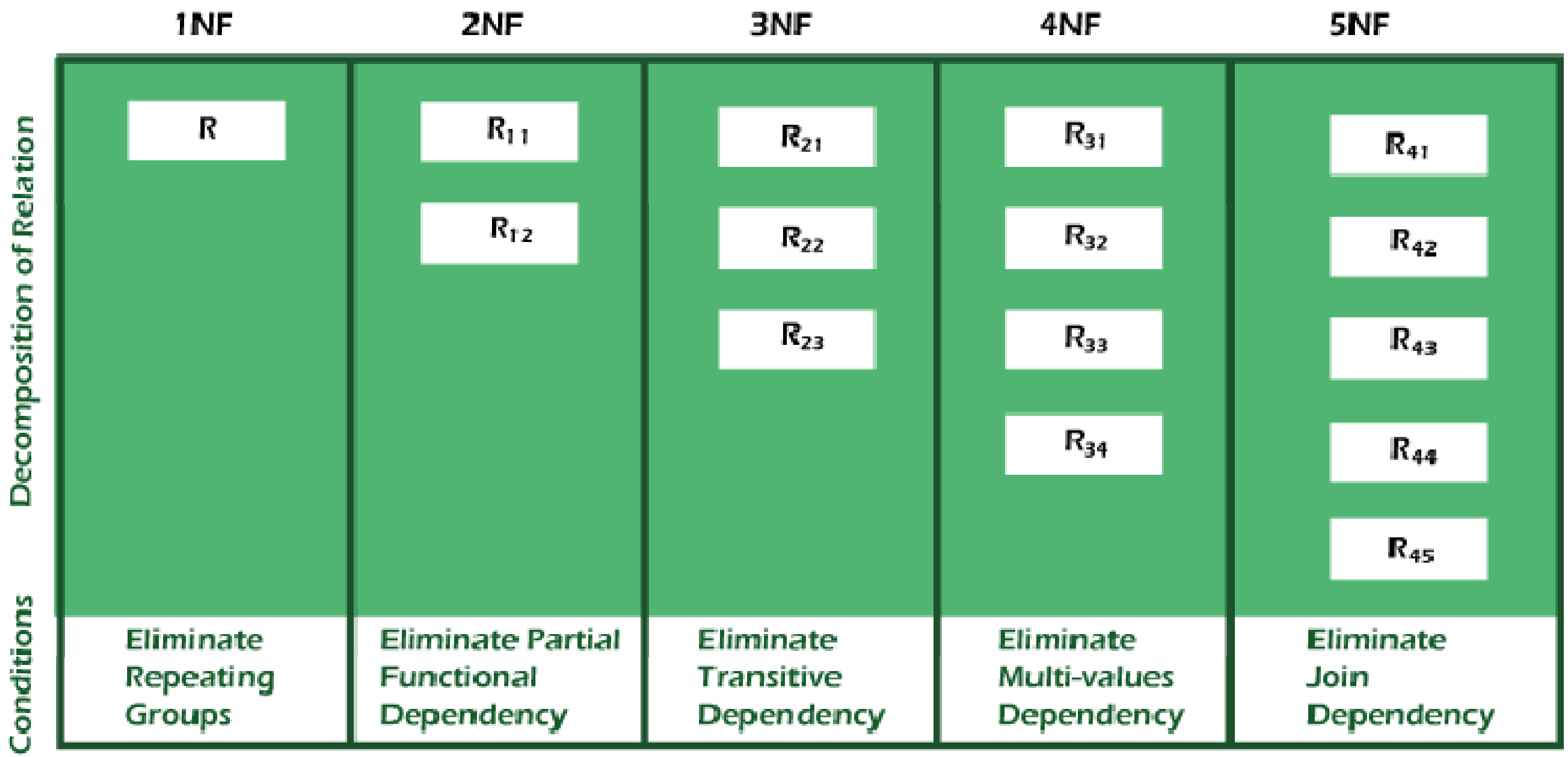
3 Normal Forms Based on Primary Keys

- 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

3.1 Normalization of Relations (1)

- **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


1NF
2NF
3NF
4NF
5NF
Decomposition of Relation
Conditions
R
R_{11}
R_{12}
R_{21}
R_{22}
R_{23}
R_{31}
R_{32}
R_{33}
R_{34}
R_{41}
R_{42}
R_{43}
R_{44}
R_{45}
Eliminate Repeating Groups
Eliminate Partial Functional Dependency
Eliminate Transitive Dependency
Eliminate Multi-values Dependency
Eliminate Join Dependency

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 (Chapter 11)
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation; Chapter 11)

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, BCNF or 4NF)
- **Denormalization**:
  - The process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

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

- A **key** 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]

Eg: {SSN},{SSN,ENAME}, {SSN,ENAME,SEX}

- 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.

Eg:{SSN,ENAME}, {SSN,ENAME,SEX}

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 **secondary keys**.

  Eg: PRIMARY KEY = {SSN}, Secondary Key ={ESSN},
- A **Prime attribute** must be a member of *some* candidate key

  Eg: SSN, PNO of Works-on {SSN,PNO}
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

- Repetition of data increases the size of database.

- Other issues like:
  - Insertion Problems
  - Deletion Problems
  - Updation Problems

Problems due to redundancy

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337

Insertion anomaly

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337
5	Ekon	CSE	Mr. X	53337

To insert redundant data for every new row (of Student data in our case) is a data insertion problem or anomaly.

Deletion Anamoly

STUDENTS TABLE

rollno	name	branch	hod	office_tel

Branch information deleted along with Student data.

Deletion Anomaly

Loss of a related dataset when some other dataset is deleted.

Updation Anamoly

STUDENTS TABLE

rollno	name	branch	hod	office_tel
1	Akon	CSE	Mr. X	53337
2	Bkon	CSE	Mr. X	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	Mr. X	53337

Mr. X leaves, and Mr. Y joins as the new HOD for CSE

STUDENTS TABLE

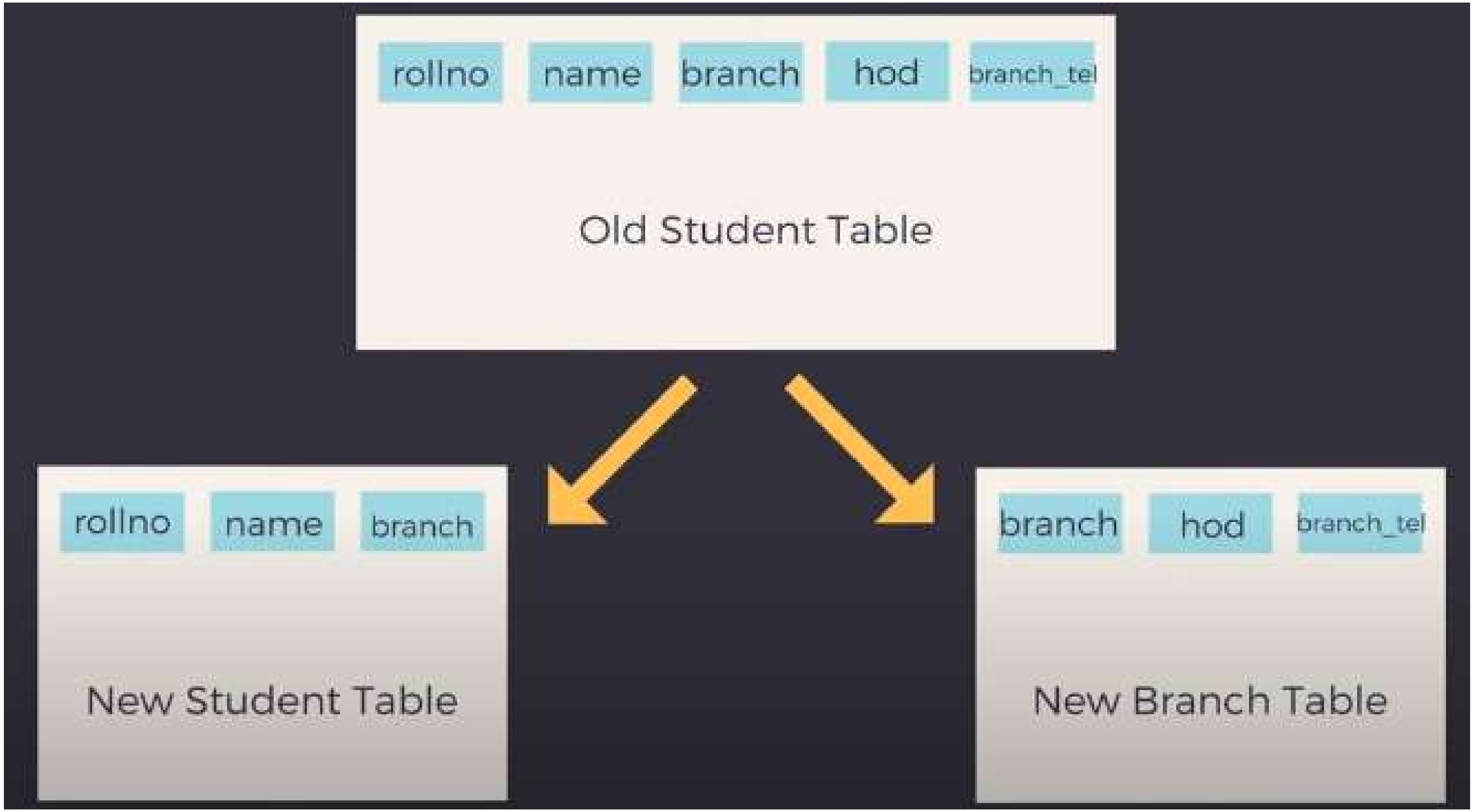
rollno	name	branch	hod	office_tel
1	Akon	CSE	~~Mr. X~~ Mr. Y	53337
2	Bkon	CSE	~~Mr. X~~ Mr. Y	53337
3	Ckon	CSE	Mr. X	53337
4	Dkon	CSE	~~Mr. X~~ Mr. Y	53337

While updating the table, if one
Of the row missed to update,
Data inconsistency can occur.

How Normalization will solve all these problems?

Student Table

Student Table + Branch Table


rollno
name
branch
hod
branch_tel
Old Student Table
rollno
name
branch
New Student Table
branch
hod
branch_tel
New Branch Table

First Normal Form

- All the attributes must have atomic values
- Disallows
  - composite attributes
  - multivalued attributes
  - **nested relations**; attributes whose values for an *individual tuple* are non-atomic

Example

- In the student table, subject column is multivalued attribute.
- Student table is not in 1NF

STUDENTS TABLE

rollno	name	subject
101	Akon	OS, CN
103	Ckon	JAVA
102	Bkon	C, C++

STUDENTS TABLE

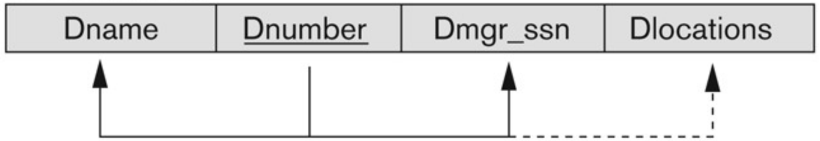
rollno	name	subject
101	Akon	OS
101	Akon	CN
103	Ckon	JAVA
102	Bkon	C
102	Bkon	C++

Normalize into 1NF

Figure 10.8 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 10.8
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

Figure 10.9 Normalization 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, AliciaJ.	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

Figure 10.9
Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Example extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

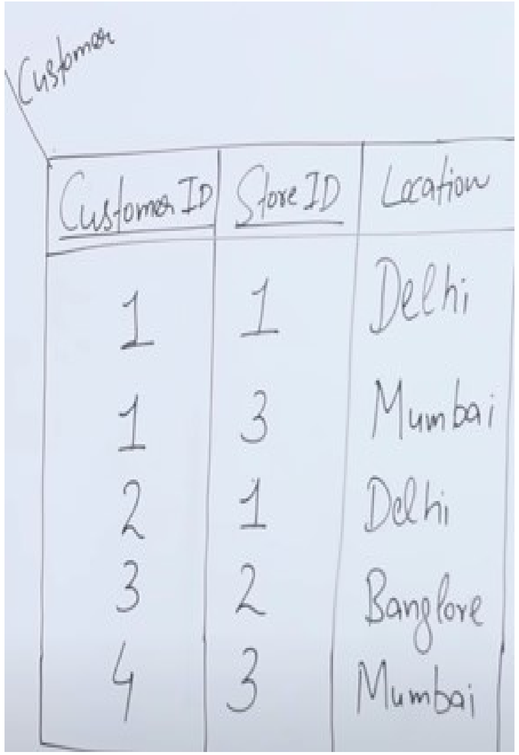
Second Normal Form (1)

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

Second Normal Form (2)

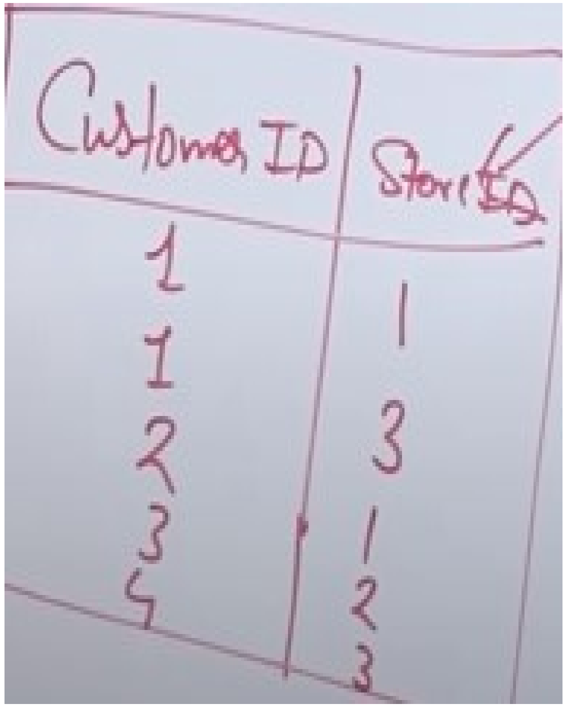
- A relation should be in 1NF.
- A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on the primary key
- R can be decomposed into 2NF relations via the process of 2NF normalization .

2NF

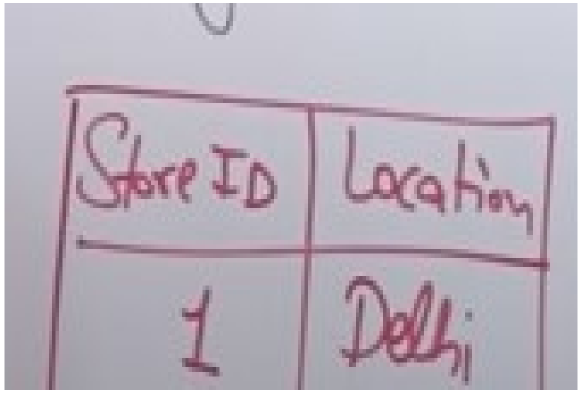


Customer

Customer ID	Store ID	Location
1	1	Delhi
1	3	Mumbai
2	1	Delhi
3	2	Banglore
4	3	Mumbai



Customer ID	Store ID
1	1
1	3
2	1
3	2
4	3



Store ID	Location
1	Delhi

Example:

- Check whether given relation is in 2NF .
- R(ABCDEF)
- FD: {C → F, E →A, EC → D, A → B}
- STEP 1: WHAT IS CANDIDATE KEYS?
- CHECK L.H.S

  C → F,
  E →A,
  EC → D,
  A → B
- EC = {FADB}
- Find closure of EC
- EC+ =ECFADB
- CK ={ EC}
- STEP2: PRIME ATTRIBUTES
- {E ,C }
- NON-PRIME ATTRIBUTES: {A,B,D,F}
- CHECK PARTIAL DEPENDANCY
- {LHS should be proper subset of CK and RHS should be a non primary key. }
- Part of candidate key determines non key attribute.
- For C → F, C is a part od CK and it determines non-key attribute. It is not in 2NF.
- E →A, FD
- A → B
- Given relation is not in 2NF.

Figure 10.10 Normalizing into 2NF and 3NF

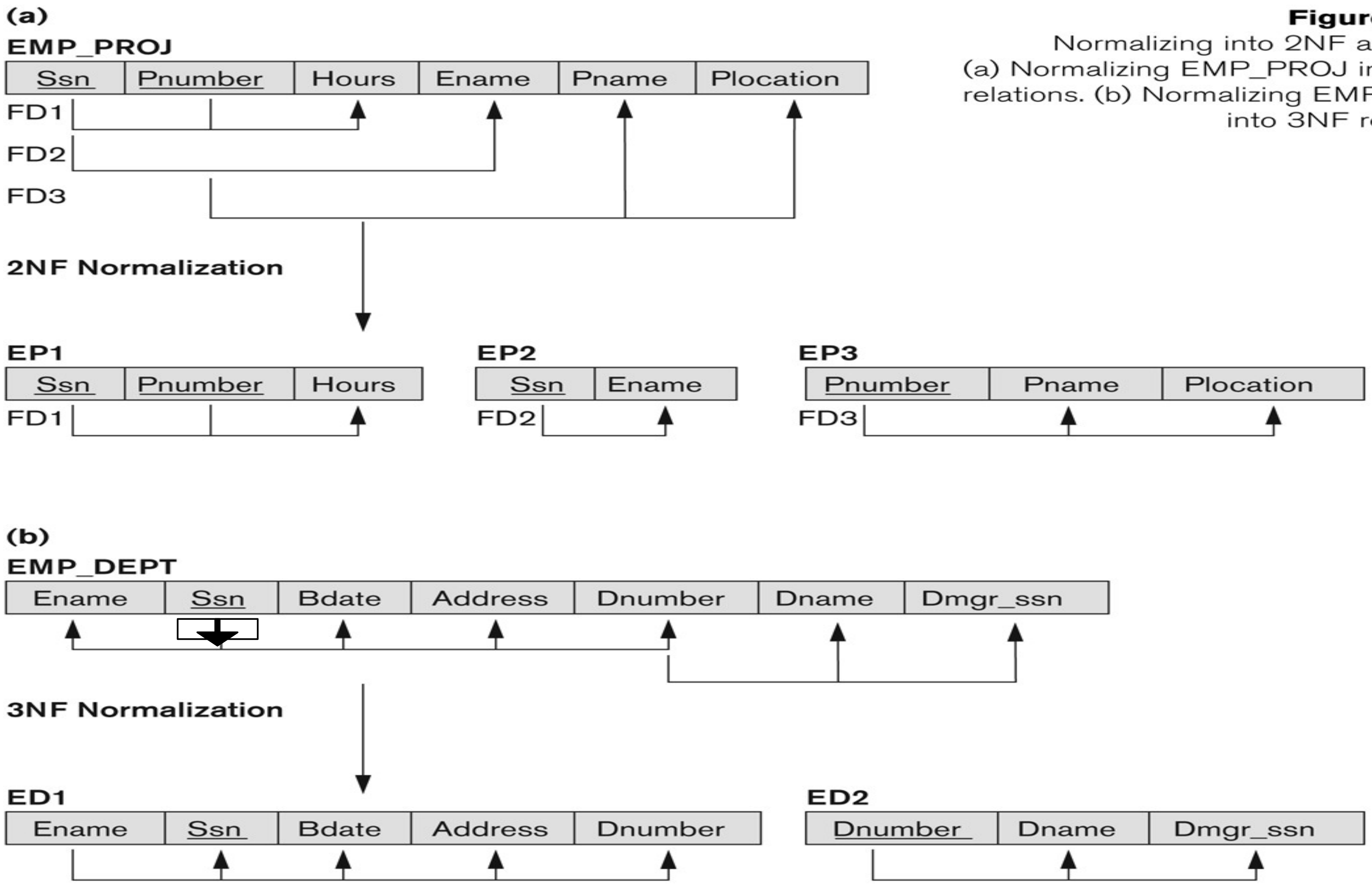


Figure 10.10
Normalizing into 2NF and 3NF. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

Figure 10.11 Normalization into 2NF and 3NF

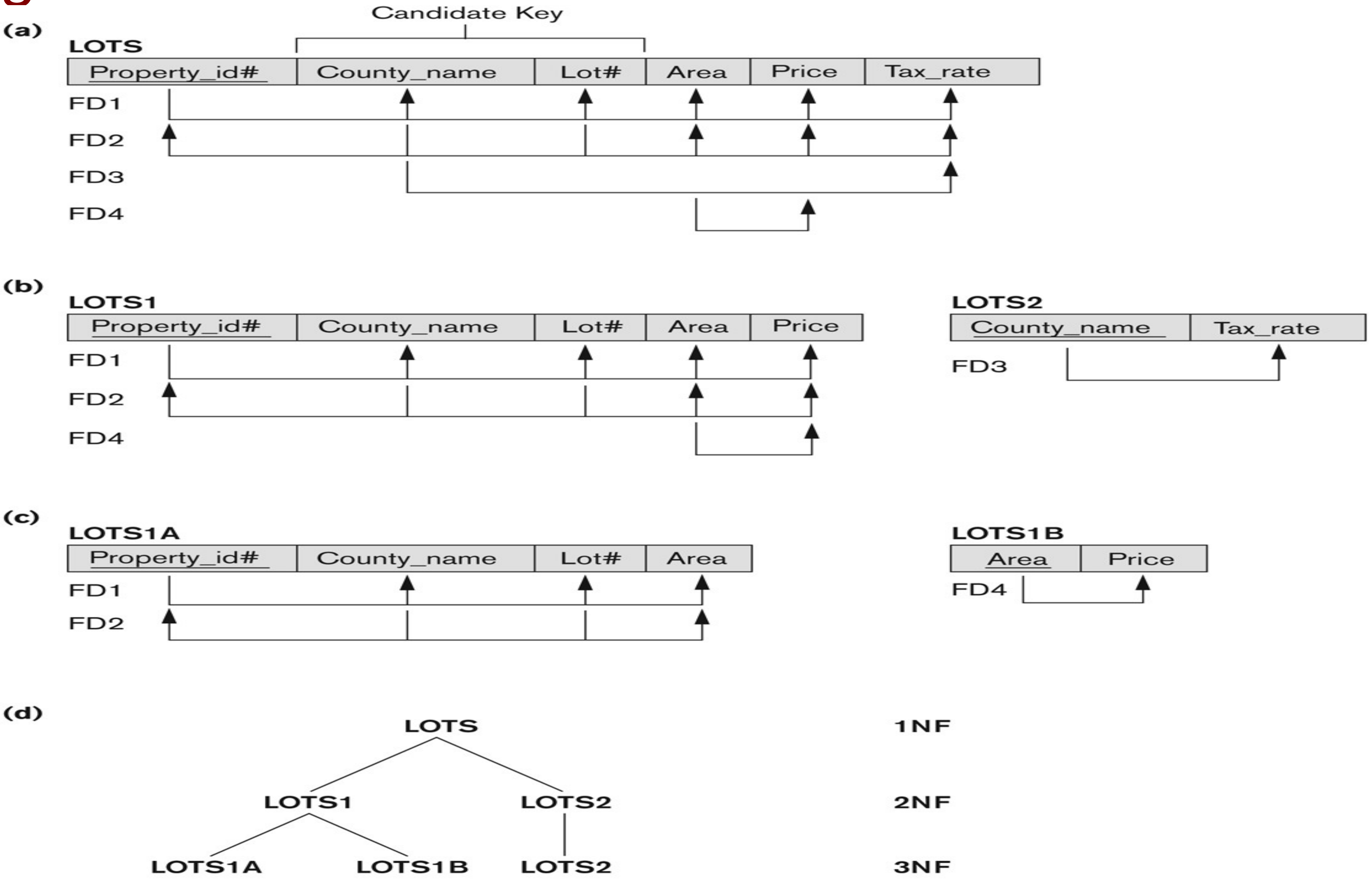


Figure 10.11
Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of the progressive normalization of LOTS.

Third Normal Form (1)

- Definition:
  - **Transitive functional dependency:** a FD X -> Z that can be derived from two FDs X -> Y and Y -> Z
- Examples:
  - 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

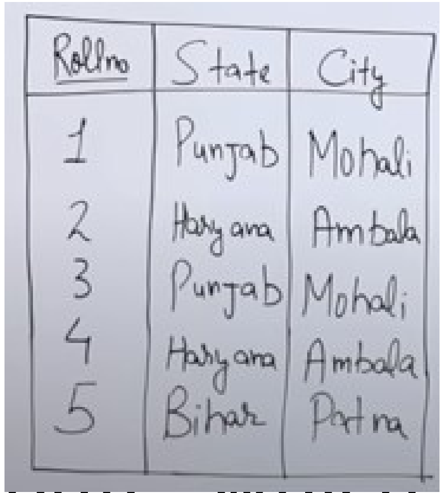
Third Normal Form (2)

- 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
- 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 part of 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.

Example 1

- Here,
- CK/PK :{rollno}
  - FD: Rollno → State

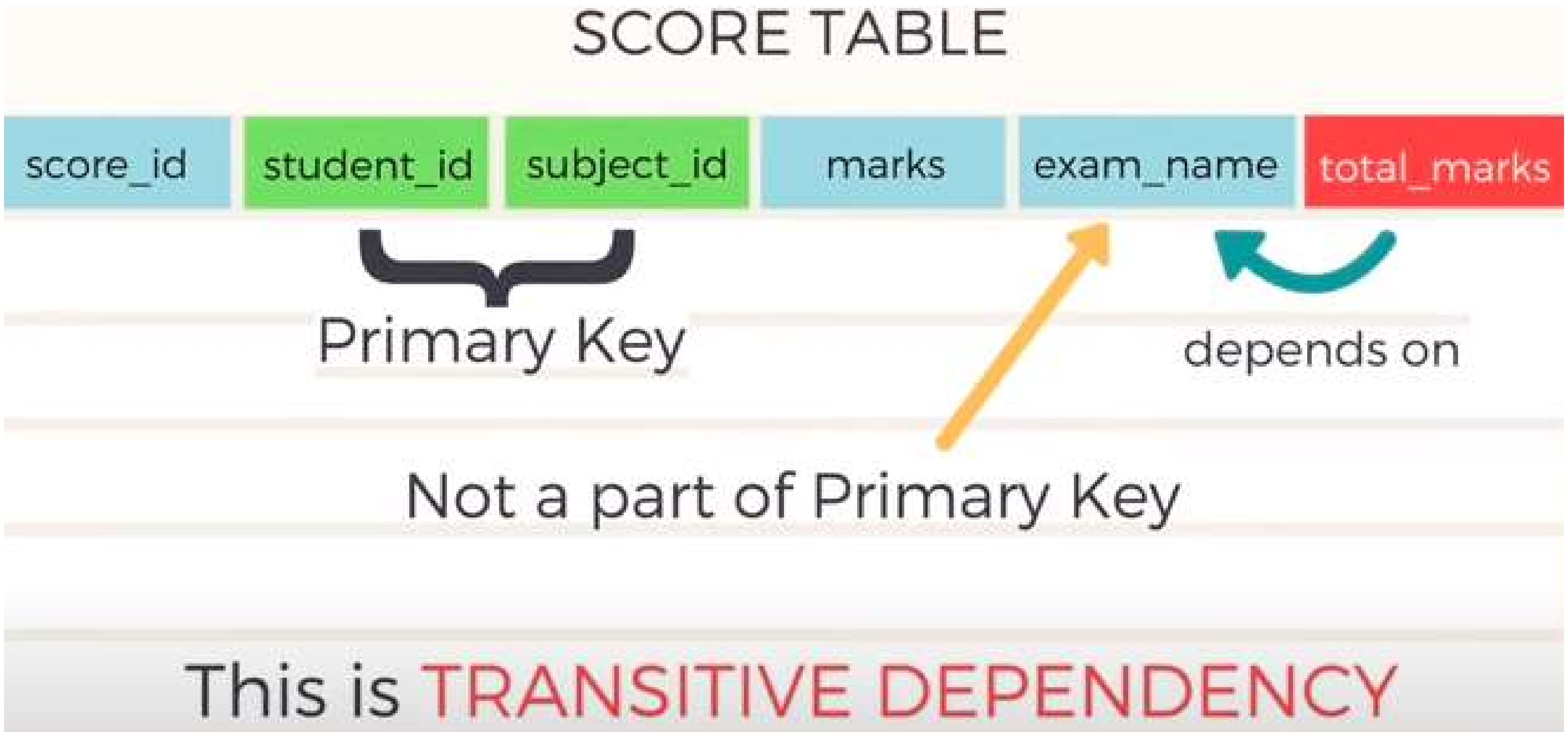
    State → City



Rollno	State	City
1	Punjab	Mohali
2	Haryana	Ambala
3	Punjab	Mohali
4	Haryana	Ambala
5	Bihar	Patna

- state is a non-prime attribute, which is trivially dependent.
- Given relation is not in 3NF.
- Decompose relation as:
- Stud(Rollno,City), City(city,state)

Example 2



- Given relation is not in 3NF

- Solution:
- Split the relation into two relations, such as-

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**

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

- The above definitions consider the primary key only
- The following more general definitions take into account relations with 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

General Normal Form Definitions
Third normal form (3NF)

- 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 -> A holds in R, then either:
    - (a) X is a key of R, or
    - (b) A is a prime attribute of R

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
- In simple terms, for any case (say, X->Y), X can't be a non-prime attribute.
- 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
- The goal is to have each relation in BCNF (or 3NF)

Example

College Enrollment Table

student_id	subject	professor
101	Java	P. Java
101	C++	P. Cpp
102	Java	P. Java2
103	C#	P. Chash
104	Java	P. Java

- 1NF

✔ **Satisfies First Normal Form**

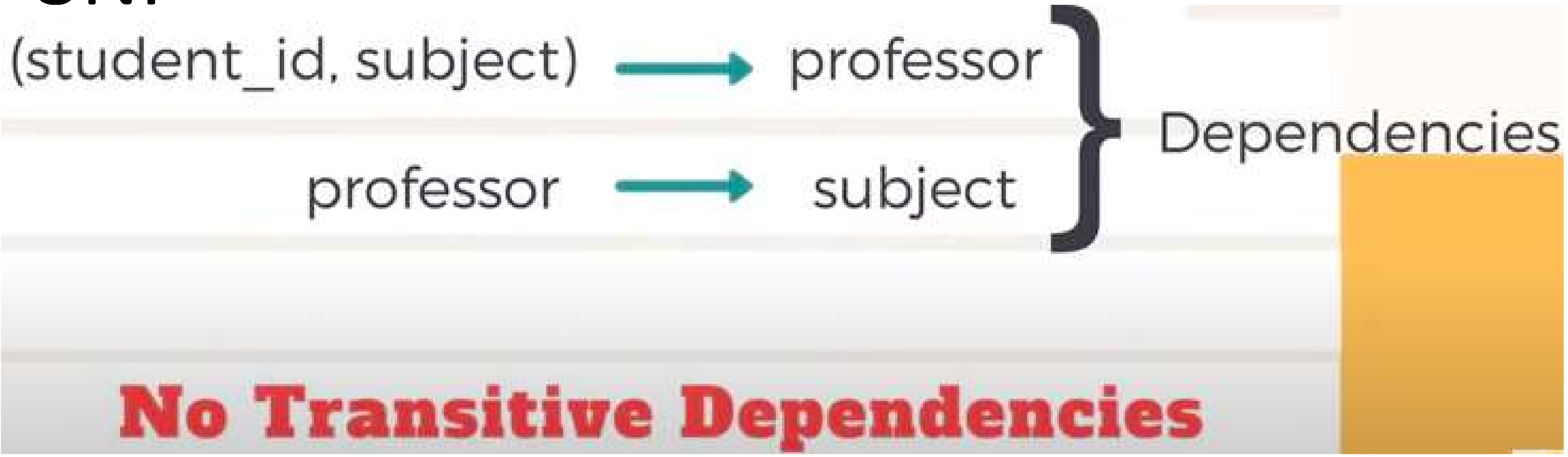
(student_id, subject) → professor

professor → subject

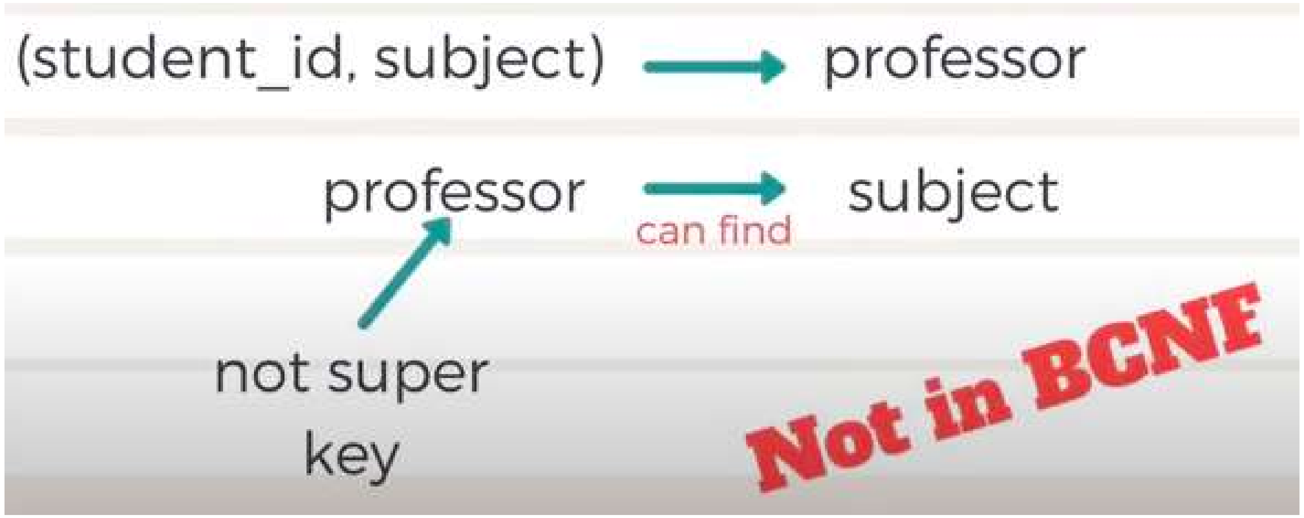
Dependencies

No Partial Dependencies

- 3NF


(student_id, subject) → professor
professor → subject
Dependencies
No Transitive Dependencies

- BCNF


(student_id, subject) → professor
professor → subject
can find
not super key
Not in BCNF

- We can decompose the table

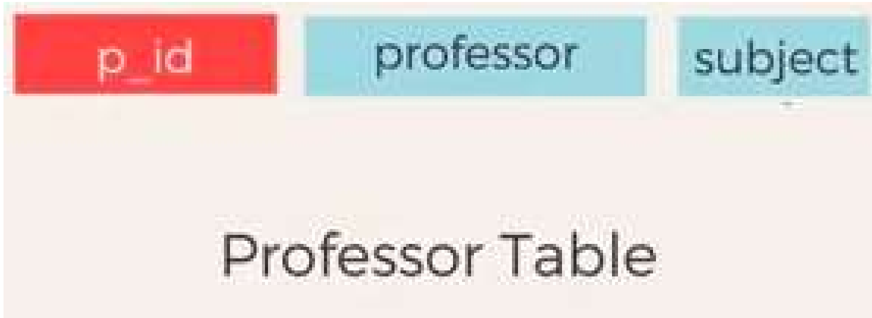


Figure 10.12 Boyce-Codd normal form

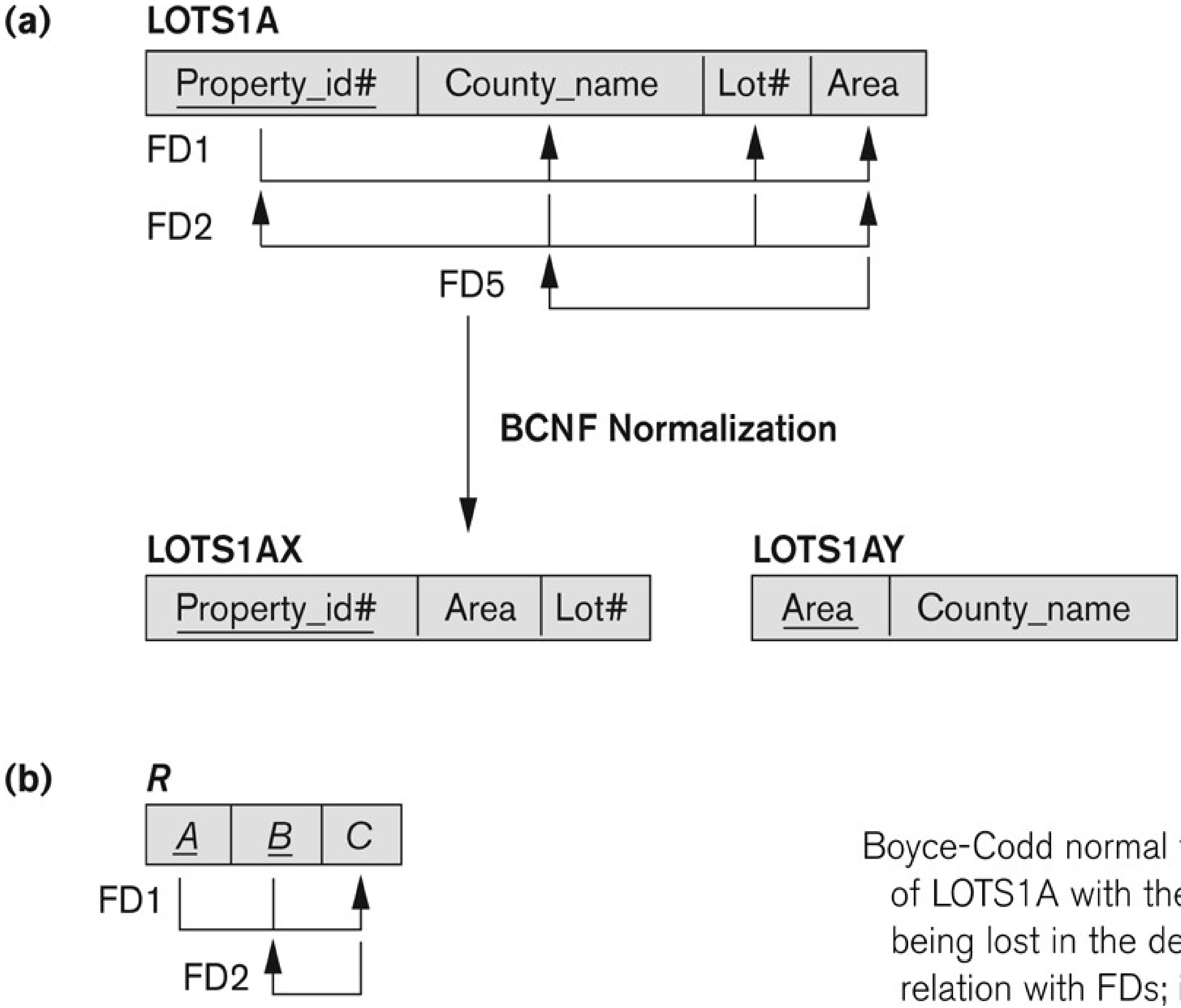


Figure 10.12
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.

Figure 10.13 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 10.13
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 10.12 (b).
  - So this 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.
  - (See Algorithm 11.3)

Chapter Outline

- Informal Design Guidelines for Relational Databases
- Functional Dependencies (FDs)
  - Definition, Inference Rules, Equivalence of Sets of FDs, Minimal Sets of FDs
- Normal Forms Based on Primary Keys
- General Normal Form Definitions (For Multiple Keys)
- BCNF (Boyce-Codd Normal Form)

Achieving the BCNF by Decomposition (2)

- Three possible decompositions for relation TEACH
  - {<u>student, instructor</u>} and {<u>student, course</u>}
  - {course, <u>instructor</u> } and {<u>course, student</u>}
  - {<u>instructor</u>, course } and {<u>instructor, student</u>}
- All three decompositions will lose fd1.
  - We have to settle for sacrificing the functional dependency preservation. But we cannot sacrifice the non-additivity 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).
- A test to determine whether a binary decomposition (decomposition into two relations) is non-additive (lossless) is discussed in section 11.1.4 under Property LJ1. Verify that the third decomposition above meets the property.

Fourth Normal Form (4NF)

- MULTIVALUED DEPENDANCY
- A → B, is multivalued dependency if

3 conditions for Multivalued Dependency

1. A→B ,for a single value of A, more than one value of B exists.
2. Tables should have at least 3 columns.

   (if table has only 2 columns, we can use 1NF to resolve it).

3. For this table with A,B,C columns, B and C should be independent.

If ALL THE 3 CONDITIONS ARE TRUE, THEN WE CAN SAY THAT THE TABLE MAY HAVE MULTI-VALUED DEPENDENCY.

EXAMPLE 1

ENROLMENT TABLE

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey

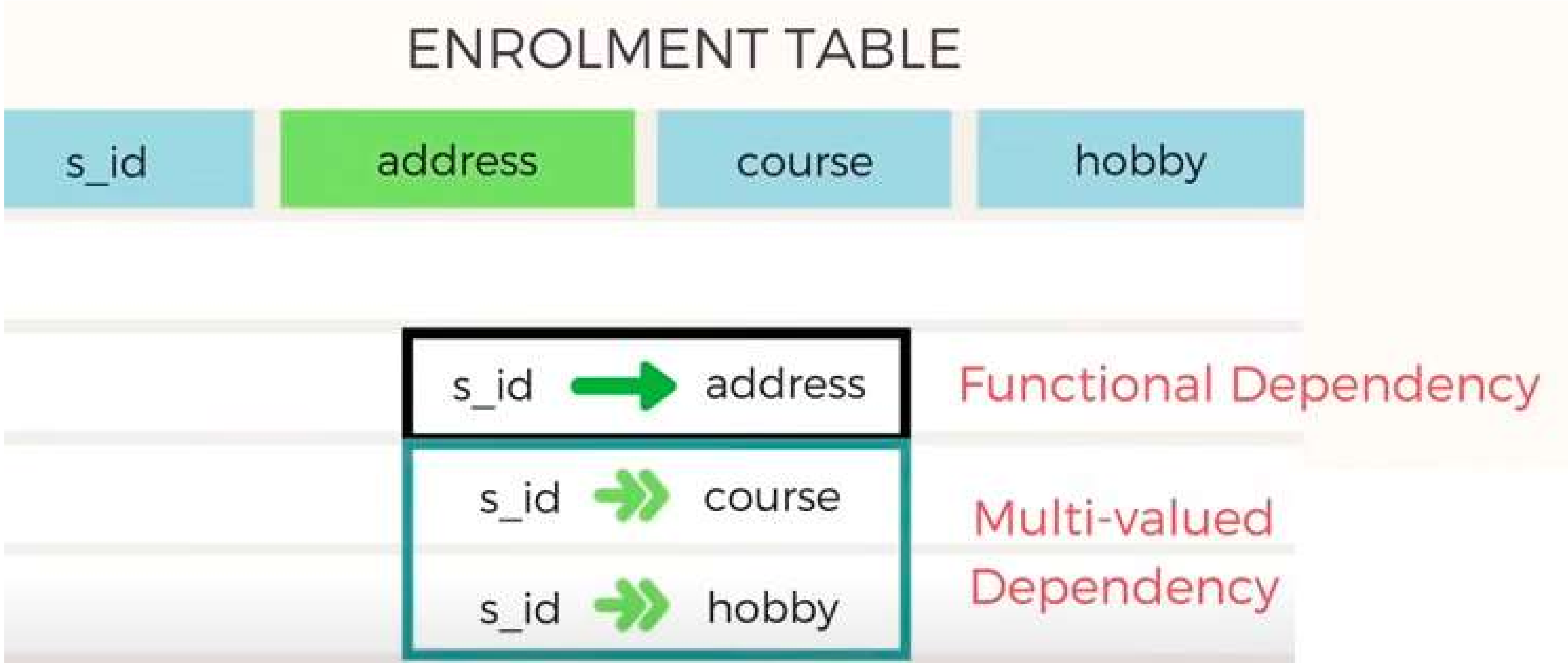
ENROLMENT TABLE

No relationship

s_id	course	hobby
1	Science	Cricket
1	Maths	Hockey
1	Science	Hockey
1	Maths	Cricket

- DECOMPOSITION of the table:

Example 2

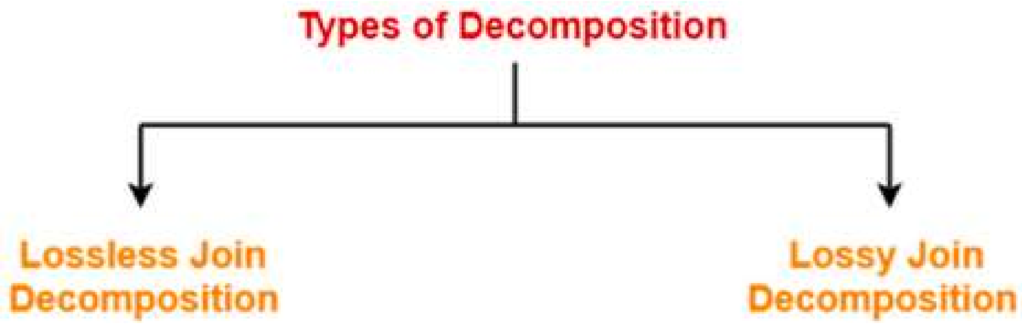

ENROLMENT TABLE
s_id
address
course
hobby
s_id → address
Functional Dependency
s_id →→ course
s_id →→ hobby
Multi-valued Dependency


Student Enrollment Table
CourseOpted Table + Hobbies Table + Address Table
(s_id & course)
(s_id & hobby)
(s_id & address)

Decomposition

- The process of decomposition in DBMS helps us remove
  - redundancy,
  - Inconsistencies,
  - anomalies from a database when we divide the table into numerous tables.
- In simpler words, the process of decomposition refers to dividing a relation X into {X1, X2,……Xn}.

Types of Decomposition



- **<u>Lossless Join Decomposition (Non-additive):</u>**
- Consider there is a relation R which is decomposed into sub relations R1 , R2 , …. , Rn.
- This decomposition is called lossless join decomposition when the join of the sub relations results in the same relation R that was decomposed.
- For lossless join decomposition, we always have

R1 $\bowtie$ R2 $\bowtie$ R3 ……. $\bowtie$ Rn = R, where $\bowtie$ is a natural join operator

- **Lossy Join Decomposition:**
- Consider there is a relation R which is decomposed into sub relations R1 , R2 , …. , Rn.
- This decomposition is called lossy join decomposition when the join of the sub relations does not result in the same relation R that was decomposed.
- The natural join of the sub relations is always found to have some extraneous tuples.
- For lossy join decomposition, we always have-
- $R1 \bowtie R2 \bowtie R3 \ldots\ldots \bowtie Rn \supset R$, where $\bowtie$ is a natural join operator

Determining Whether Decomposition Is Lossless Or Lossy

- Consider a relation R is decomposed into two sub relations R1 and R2. Then,
- If all the following conditions satisfy, then the decomposition is lossless.
- If any of these conditions fail, then the decomposition is lossy.

- **Condition-01**
- Union of both the sub relations must contain all the attributes that are present in the original relation R.
- Thus, $R1 \cup R2 = R$

Cntd…

- **Condition-02**
- Intersection of both the sub relations must not be null.
- In other words, there must be some common attribute which is present in both the sub relations.
- Thus, $R1 \cap R2 \neq \emptyset$
- **Condition-03**
- Intersection of both the sub relations must be a super key of either R1 or R2 or both.
- Thus, $R1 \cap R2 =$ Super key of R1 or R2

Properties of Decomposition

- **Lossless:**
- All the decomposition that we perform in Database management system should be lossless.
- All the information should not be lost while performing the join on the sub-relation to get back the original relation. It helps to remove the redundant data from the database.
- **Dependency Preservation:**
- Dependency Preservation is an important technique in database management system.
- It ensures that the functional dependencies between the entities is maintained while performing decomposition.
- It helps to improve the database efficiency, maintain consistency and integrity.
- **Lack of Data Redundancy:**
- Data Redundancy is generally termed as duplicate data or repeated data.
- This property states that the decomposition performed should not suffer redundant data.
- It will help us to get rid of unwanted data and focus only on the useful data or information.

Dependency preservation example