

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

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)

Chapter Outline

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

Introduction

- What is relational database design?
 - The grouping of attributes to form "good" relation schemas
- Two levels of relation schemas
 - The **logical** "user view" **level** (or conceptual level)
 - Better schema enables users to understand the meaning of the data in the relations -> formulate query correctly
 - The **storage** "base relation" **level** (Implementation level)
 - Tuples in a base relation are stored and updated correctly (anomaly-free) -> applies only to schemas of base relations
- Design is concerned mainly with base relations
- What are the criteria for "good" base relations?

Introduction (cont'd.)

- Approaches to database design:
 - Bottom-up design methodology (design by synthesis)
 - Consider the basic relationships among individual attributes as the starting point and use those to construct relation schemas
 - Not very popular in practice because it suffers from the problem of having to collect a large number of binary relationships among attributes
 - Top-down design methodology (design by analysis)
 - Starts from a number of groupings of attributes into relations that exist together naturally. The relations are then analyzed individually and collectively, leading to further decomposition until all desirable properties are met
 - The theory of Ch. 14 is applicable to both the top-down and bottom-up approaches

Introduction (cont'd.)

- The **implicit goals** of the relational database design activity are
 - **Information preservation**
 - Maintaining all concepts captured in the conceptual design, such as EER model, when mapping to the logical design
 - **Minimum redundancy**
 - Minimize the redundant storage of the same information
 - Reduce the need for multiple updates to **maintain consistency** across multiple copies of the same information

Informal Design Guidelines for Relational Databases

- 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 Normal Form)
 - - BCNF (Boyce-Codd Normal Form)
- Additional types of dependencies, further normal forms, relational design algorithms by synthesis are discussed in Chapter 15

Informal Design Guidelines for Relation Schemas

- Measures of quality
 - Making sure attribute semantics are clear
 - Reducing redundant information in tuples
 - Reducing NULL values in tuples
 - Disallowing possibility of generating **spurious tuples** (假造的資料)

Imparting Clear Semantics to Attributes in Relations

- **Semantics of a relation**
 - Meaning resulting from interpretation of attribute values in a tuple
- Easier to explain semantics of relation
 - Indicates better schema design

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

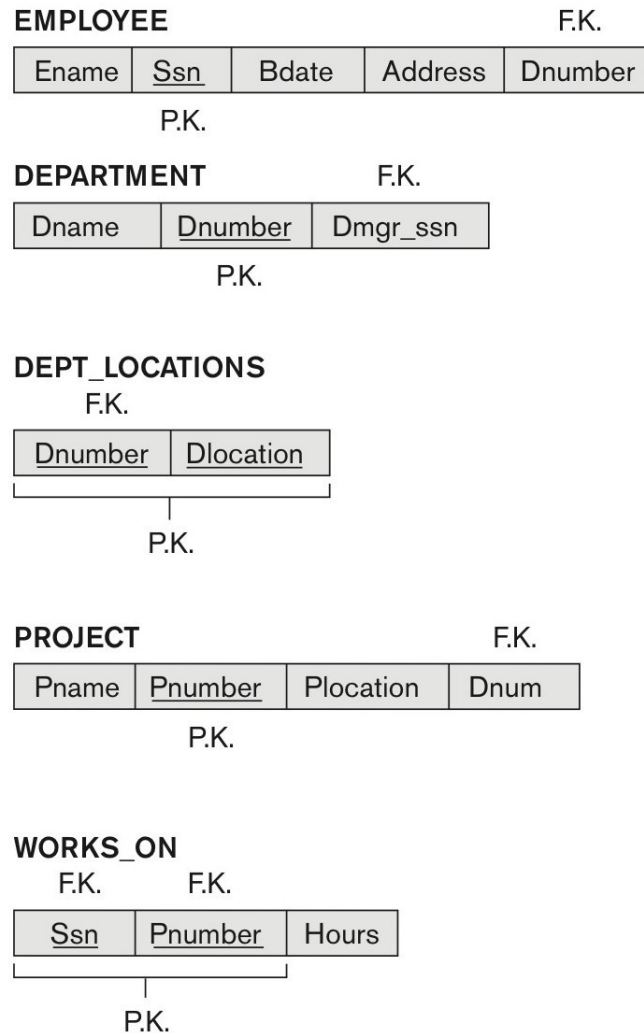


Figure 14.1 A simplified COMPANY relational database schema.

Figure 14.2 Sample database state for the relational database schema in Figure 14.1.

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	291 Berry, 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

Guideline for Imparting Clear Semantics to Attributes in Relations

■ Guideline 1

- Design relation schema so that it is easy to explain its meaning
- Do not combine attributes from multiple entity types and relationship types into a single relation
- Intuitively, if a relation schema corresponds to one entity type or one relationship type, it is straightforward to explain its meaning.
- Otherwise, if the relation corresponds to a mixture of multiple entities and relationships, semantic ambiguities will result and relation cannot be easily explained

1.2 Redundant Information in Tuples and Update Anomalies

- Information is stored redundantly
 - Wastes storage
 - For examples, see Figure 14.4
 - EMP_DEPT vs. Employee + Department
 - EMP_PRJ vs. Employee + Project
 - Causes problems with **update anomalies**
 - Insertion anomalies
 - Deletion anomalies
 - Modification anomalies
 - Storing natural joins of base relations leads to **update anomalies**

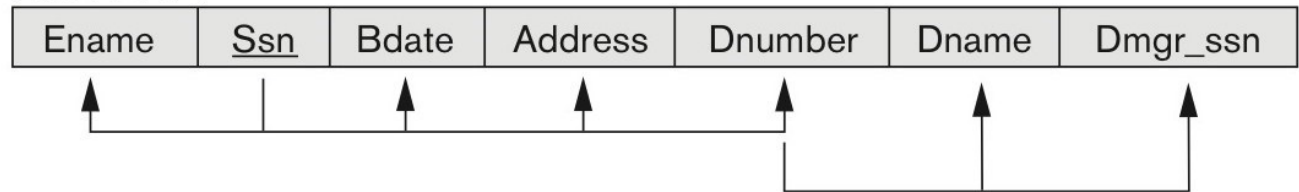
Figure 14.3 Two relation schemas suffering from update anomalies

Figure 14.3

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

(a)

EMP_DEPT



(b)

EMP_PROJ

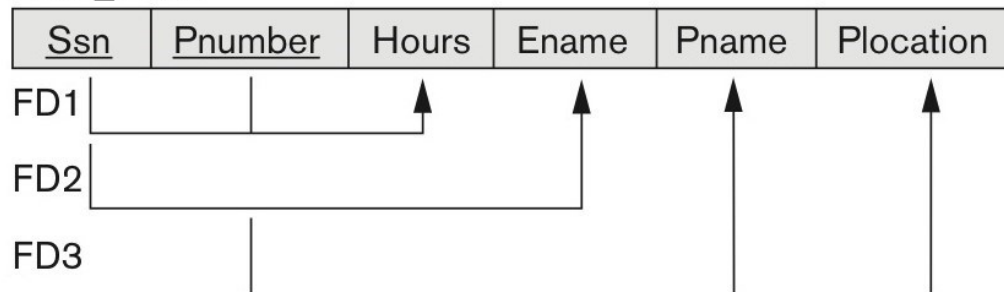


Figure 14.4 Sample states for EMP_DEPT and EMP_PROJ

Figure 14.4

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	<u>Ssn</u>	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

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

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.
 - The inserted value for Pname must **be consistent** with the values in other tuples that have the same Proj#
- Conversely
 - Cannot insert an employee unless a he/she is assigned to a project.
 - **NULL value** in Proj# (primary key) can cause problem

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.

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. Otherwise, the database will become *inconsistent*

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 clearly so that applications can be made to take them into account.

1.3 Null Values in Tuples

- May group many attributes together into a “fat” relation
 - Can end up with many NULLs
- Problems with NULLs
 - Wasted storage space
 - e.g., if only 10% employees have offices, including attribute officeNum in EMP will waste space -> **separate**
 - Problems understanding meaning
 - Null can be
 - Attribute not applicable or invalid
 - Attribute value unknown (may exist)
 - Value known to exist, but unavailable (absent)
 - The results become unpredictable with JOIN and aggregation operations

Guideline for Null Values in Tuples

■ GUIDELINE 3:

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

Generation of Spurious Tuples

- Figure 14.5(a)
 - Relation schemas EMP_LOCS and EMP_PROJ1 (instead on EMP_PROJ)
 - **Plocation** is neither a primary key nor a foreign key in either EMP_LOCS or EMP_PROJ1
- NATURAL JOIN
 - Result produces many **more tuples** than the original **set of tuples** in EMP_PROJ
 - Called **spurious tuples**
 - Represent spurious information that is **not valid**

Figure 14.5 Particularly poor design for the EMP_PROJ relation in Figure 14.3(b). (a) The two relation schemas EMP_LOCS and EMP_PROJ1. (b) The result of projecting the extension of EMP_PROJ from Figure 14.4 onto the relations EMP_LOCS and EMP_PROJ1.

(a)

EMP_LOCS

Ename	Plocation
-------	-----------

P.K.

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
-----	---------	-------	-------	-----------

P.K.

(b)

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston

Figure 14.6 Result of applying NATURAL JOIN to the tuples in EMP_PROJ1 and EMP_LOCS of Figure 14.5 just for employee with Ssn = "123456789". Generated spurious tuples are marked by asterisks.

Ssn	Pnumber	Hours	Pname	Plocation	Ename
123456789	1	32.5	ProductX	Bellaire	Smith, John B.
* 123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
123456789	2	7.5	ProductY	Sugarland	Smith, John B.
* 123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
* 123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
* 666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
* 453453453	1	20.0	ProductX	Bellaire	Smith, John B.
453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
* 453453453	2	20.0	ProductY	Sugarland	Smith, John B.
453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
* 453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
* 333445555	2	10.0	ProductY	Sugarland	Smith, John B.
* 333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
* 333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
* 333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.
333445555	20	10.0	Reorganization	Houston	Wong, Franklin T.

* *spurious*
tuples(假的;
偽造的)

*
*
*

1.4 Generation of Spurious Tuples – avoid at any cost

- 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 (or nonadditive) join** condition.
 - **No spurious tuples** should be generated by doing a natural-join of any relations.

Guideline for Generation of Spurious Tuples

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

Spurious Tuples (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 and Discussion of Design Guidelines

- **Anomalies cause redundant work to be done or loss of information**
 - Insertion anomaly
 - Modification anomaly
 - Deletion anomaly (cause the loss of information)
- **Waste of storage space due to NULLs**
- **Difficulty of performing operations and joins due to NULL values**
- **Generation of invalid and spurious data during joins **due to improper relationship** (foreign key, primary key) **of matched attributes****

2. Functional Dependencies

- **Functional dependencies (FDs)**
 - Are used to specify *formal measures* of the "goodness" of relational designs
 - Enables us to detect and describe some of the above-mentioned problems in precise terms
 - And keys are used to define **normal forms** for relations
 - Important concept in relational schema design theory
 - Theory of functional dependency
 - Are **constraints** that are derived from the meaning and interrelationships of the data attributes

2.1 Defining Functional Dependencies

- Constraint between two sets of attributes from the database

Definition. A functional dependency, denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of R specifies a *constraint* on the possible tuples that can form a relation state r of R . The constraint is that, for any two tuples t_1 and t_2 in r that have $t_1[X] = t_2[X]$, they must also have $t_1[Y] = t_2[Y]$.

- **Property of semantics or meaning of the attributes**
 - Are constraints that are derived from the meaning and interrelationships of the data attributes in real world
- **Legal relation states**
 - Satisfy the functional dependency constraints (*hold at all times*)

2.1 Defining Functional Dependencies (cont'd.)

- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y
 - Y is *functionally dependent on* X
 - X is called the left-hand side of the FD (or f.d.)
 - Y is called the right-hand side of the FD (or f.d.)

2.1 Defining Functional Dependencies (cont'd.)

- $X \rightarrow Y$ 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
- Given a populated relation
 - Cannot determine which FDs hold and which do not
 - Unless meaning of and relationships among attributes known
 - Can state that FD does not hold if there are tuples that show violation of such an FD

Examples of FD constraints (1)

- Social security number determines employee name
 - $SSN \rightarrow ENAME$
- Project number determines project name and location
 - $PNUMBER \rightarrow \{PNAME, PLOCATION\}$
- Employee ssn and project number determines the hours per week that the employee works on the project
 - $\{SSN, PNUMBER\} \rightarrow HOURS$

Examples of FD constraints (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* \rightarrow *Course* may exist. However, the FDs *Teacher* \rightarrow *Course*, *Teacher* \rightarrow *Text* and *Couse* \rightarrow *Text* are ruled out.

An FD cannot be inferred automatically from a given relation extension *r* but must be defined explicitly by someone who knows the semantics of the attributes of *R*

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 $R(A, B, C, D)$ 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

May hold

$B \rightarrow C$; $C \rightarrow B$; $\{A, B\} \rightarrow C$; $\{A, B\} \rightarrow D$; $\{C, D\} \rightarrow B$

Do NOT hold (can be inferred from the relation state)

$A \rightarrow B$; $B \rightarrow A$; $D \rightarrow C$

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**

Normalization for Relational Schema Design

- Approaches for relational schema design
 - Perform a conceptual schema design using a conceptual model then map conceptual design into a set of relations
 - Design relations based on external knowledge derived from existing implementation of files or forms or reports
- Following either of these approaches, it is then useful to evaluate the relations for goodness and decompose them further as needed to achieve higher normal forms, using the normalization theory

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 may be needed to ensure a good relational design (**lossless join**, **dependency preservation**; see Chapter 15)

Normalization of Relations (3)

- First proposed by Codd in 1972
 - First normal form (1NF), second normal form 2NF), and third normal form (3NF)
- Takes a relation schema through a series of tests
 - Certify whether it satisfies a certain normal form
 - Proceeds in a top-down fashion
 - *Relational design by analysis*
- **Normalization of data**
 - A process of analyzing the given relation schemas based on their FDs and primary keys to achieve
 - Minimizing redundancy
 - Minimizing the insertion, deletion, and update anomalies

Normalization of Relations (4)

- The normalization procedure provides database designers with the following:
 - A **formal framework** for analyzing relation schemas based on their keys and on the functional dependencies among their attributes
 - A series of **normal form test** that can be carried out on individual relation schemas so that the relational database can be normalized to any desired degree
- **Normal form definition**

Definition. The normal form of a relation refers to the highest normal form condition that it meets, and hence indicates the degree to which it has been normalized.

Normalization of Relations (5)

- The normal form **does not guarantee** a good DB design **if it is considered in isolation**
- **Additional properties** may be needed to ensure a good relational design
- Properties that the relational schemas should have:
 - **Nonadditive join property**
 - Extremely critical and *must be achieved at any cost*
 - **Dependency preservation property**
 - Desirable but sometimes sacrificed for other factors

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 (above 4NF) 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**
 - for improving performance

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

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

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**.
- 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.

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
- Only attribute values permitted are single **atomic (or indivisible) values**
- Most RDBMSs allow **only** those relations to be defined that are in First Normal Form

Figure 14.9 Normalization into 1NF

(a)

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn	Dlocations



(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.

3.4 First Normal Form (cont'd.)

- Techniques to achieve first normal form
 - Remove attribute and place in separate relation along with the primary key
 - Expand the key
 - *has the disadvantage of introducing redundancy*
 - Use several atomic attributes (if the maximum number of values is known for the attribute)
 - *disadvantage of introducing NULL values*
 - *Introduce spurious semantics about the ordering among the values of the multivalued attributes that is not originally intended*
 - *Querying on this attribute becomes more difficult*

3.4 First Normal Form (cont'd.)

- Does not allow **nested relations** (i.e., multivalued composite attribute)
 - Each tuple can have a relation **within** it
 - Ex:
 - EMP_PROJ(Ssn, Ename, {PROJS(Pnumber, Hours)})
 - Pnumber is the **partial key** of the nested relation PROJS
- To change to 1NF:
 - Remove nested relation attributes into a new relation
 - Propagate the primary key into it
 - **Unnest** relation into a set of 1NF relations

Figure 14.10 Normalizing nested relations into 1NF

(a)

EMP_PROJ		Projs	
Ssn	Ename	Pnumber	Hours

(b)

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
-----	---------	-------

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.

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*
 - $A \in X$ and $(X - \{A\})$ does NOT functionally determine Y
- A functional dependency $X \rightarrow Y$ is a **partial dependency** if some attribute $A \in X$ can be removed from X and the dependency still holds
- Examples:
 - $\{SSN, PNUMBER\} \rightarrow HOURS$ is a full FD since neither $SSN \rightarrow HOURS$ nor $PNUMBER \rightarrow HOURS$ hold
 - $\{SSN, PNUMBER\} \rightarrow ENAME$ is not a full FD (it is called a partial dependency) since $SSN \rightarrow ENAME$ also 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
- The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key
 - If the primary key contains a single attribute, the test need not be applied at all
- R can be decomposed into 2NF relations via the process of 2NF normalization or “second normalization”
 - Second normalize into a number of 2NF relations
 - Nonprime attributes are associated only with part of primary key on which they are fully functionally dependent

Figure 14.11 Normalizing into 2NF and 3NF

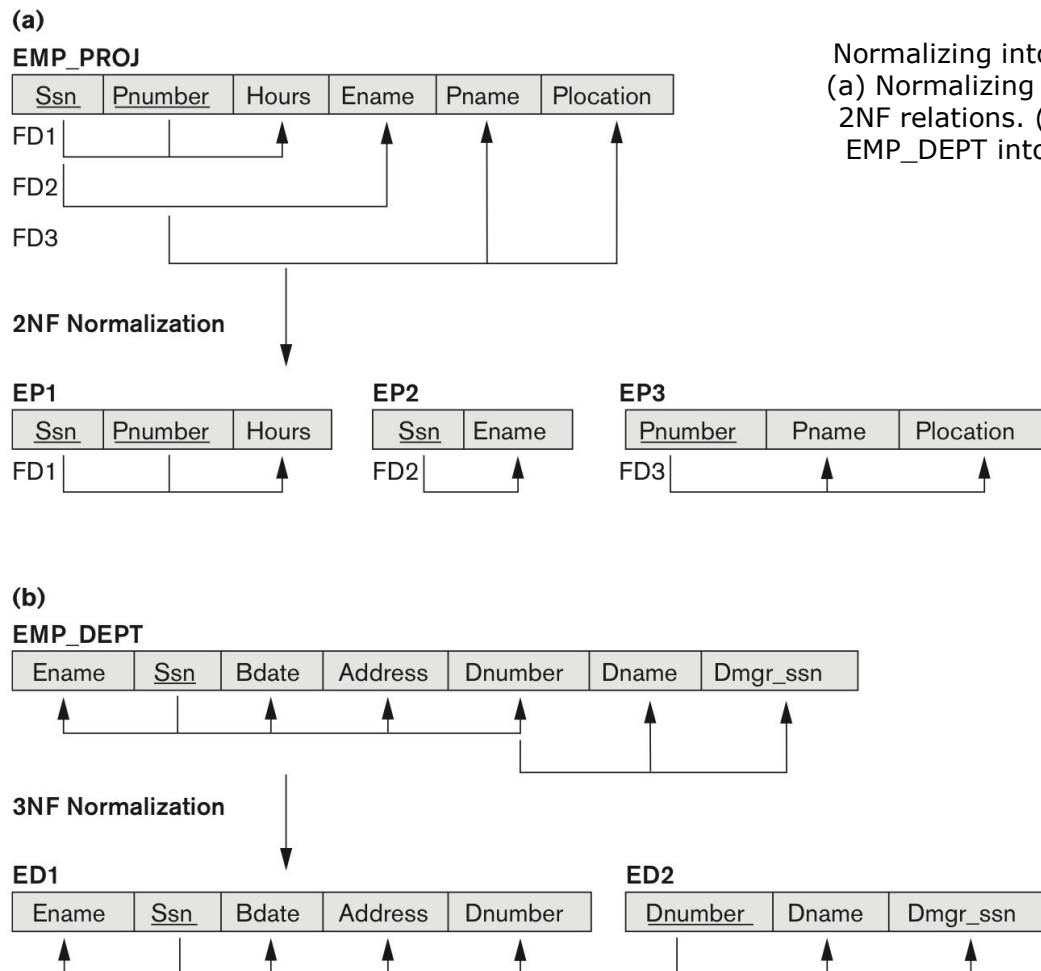


Figure 14.11
 Normalizing into 2NF and 3NF.
 (a) Normalizing EMP_PROJ into 2NF relations.
 (b) Normalizing EMP_DEPT into 3NF relations.

3.6 Third Normal Form (1)

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

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 $X \rightarrow Y$ 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, $SSN \rightarrow Emp\# \rightarrow Salary$ and Emp# is a candidate key.

Third Normal Form (3)

- Problematic FD
 - Left-hand side is **part of primary key**
 - Left-hand side is a **nonkey attribute**
 - 2NF and 3NF normalization remove these problem FDs by decomposing the original relation into new relations

Normal Forms Defined Informally

- 1st normal form
 - All attributes depend on **the key**
- 2nd normal form
 - All attributes depend on **the whole key**
 - No partial dependencies on the primary key (for nonprime attributes)
- 3rd normal form
 - All attributes depend on **nothing but the key**
 - No transitive dependencies on the primary key (for nonprime attribute)

General Definitions of Second and Third Normal Forms

*In addition to the primary key, **take all candidate keys of a relation into account***

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).

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

- We want to design our relation schemas so that they have neither partial nor transitive dependencies because these types of dependencies cause the **update anomalies**
- 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**.

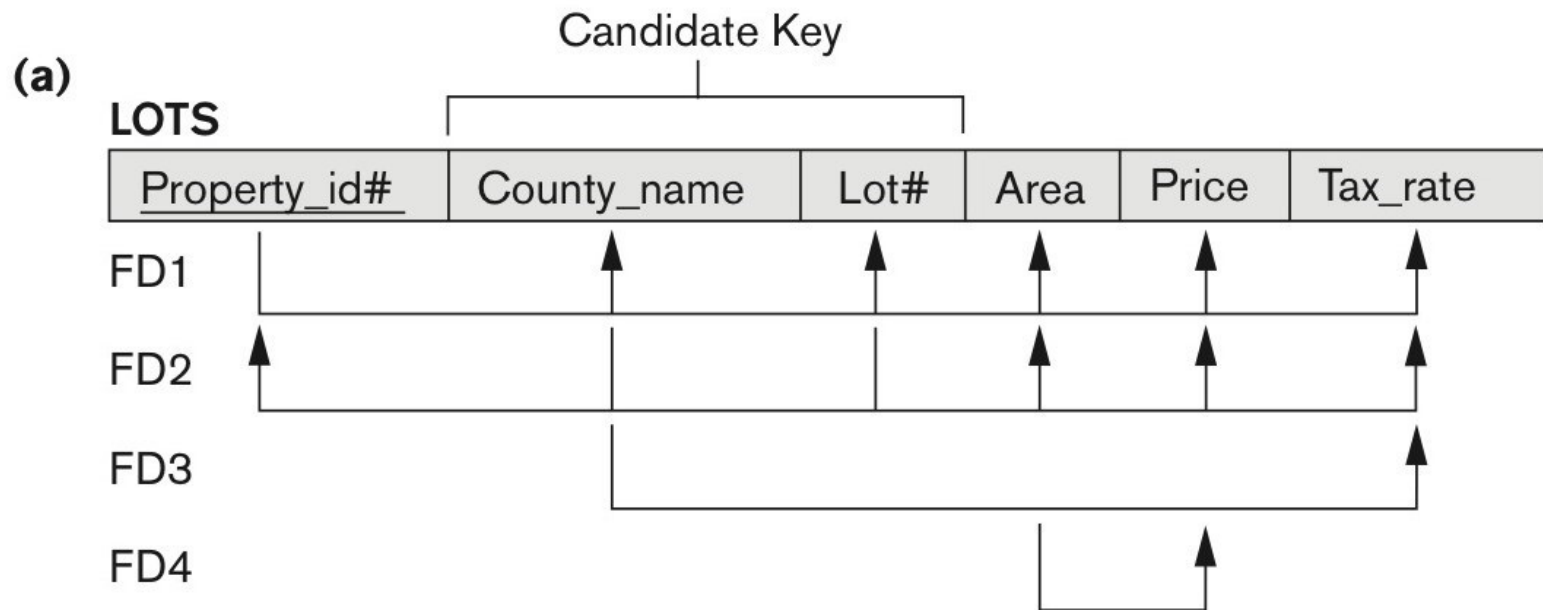
4.1 General Definition of 2NF (For Multiple Candidate Keys)

- 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 FD
 $\text{County_name} \rightarrow \text{Tax_rate}$ violates 2NF.

So second normalization converts LOTS into
LOTS1 (Property_id#, County_name, Lot#, Area, Price)
LOTS2 (County_name, Tax_rate)

General Definition of Second Normal Form

Definition. A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is not partially dependent on any key of R .¹¹



FD3 causes the violation of 2NF
FD4 causes the violation of 3NF

Figure 14.12 Normalization into 2NF and 3NF

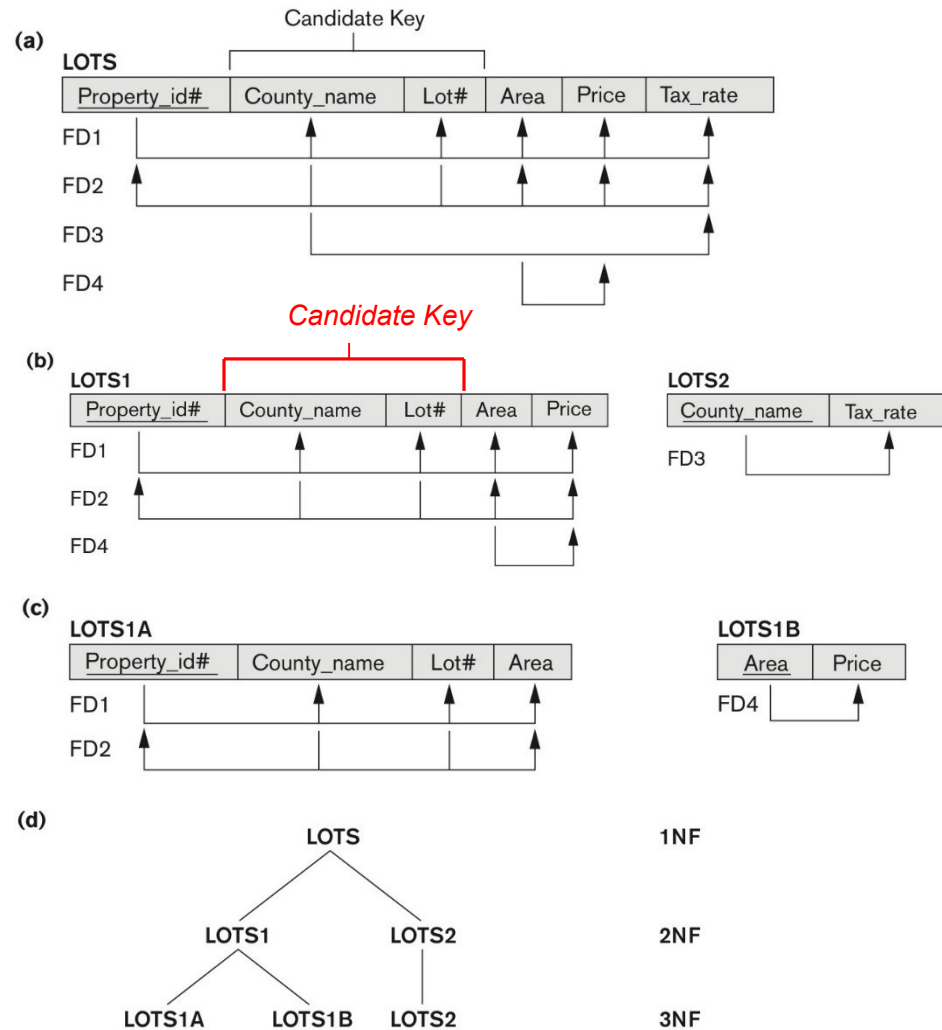
Figure 14.12

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.



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 FD4 violates 3NF because Area \rightarrow Price ; and Area is not a superkey in LOTS1. (see Figure 14.12b).
- **NOTE: Boyce-Codd normal form** disallows condition (b) above
- The general definition can be **applied directly** to test whether a relation schema is in 3 NF; it does not have to go through 2 NF first

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 $X \rightarrow A$ 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.

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 but are “caught by” BCNF which we discuss next.
 - BCNF disallows condition (b)

5. BCNF (Boyce-Codd Normal Form)

- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever an **FD $X \rightarrow A$** 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)

Figure 14.13 Boyce-Codd normal form

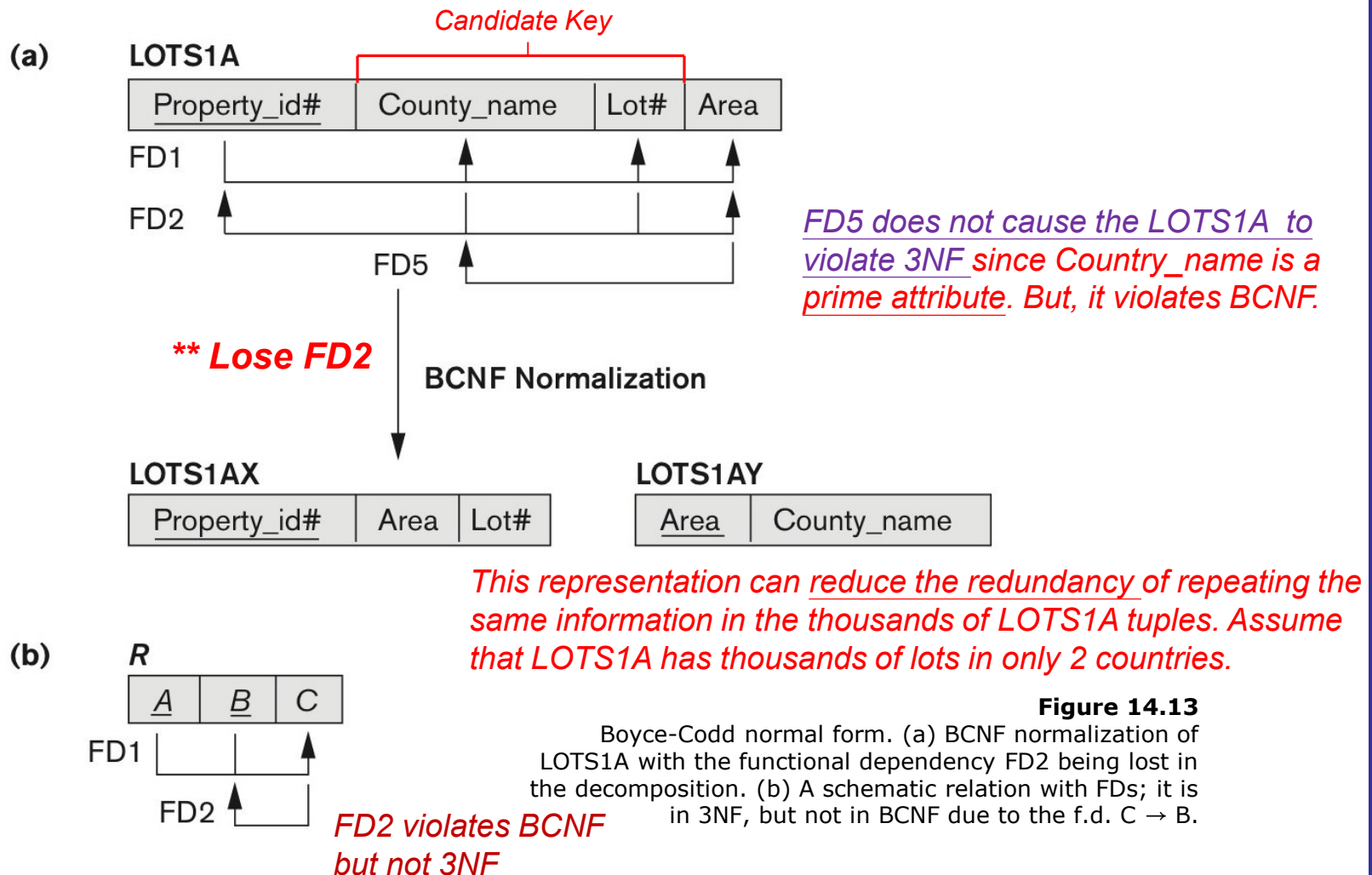


Figure 14.13
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF due to the f.d. $C \rightarrow B$.

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

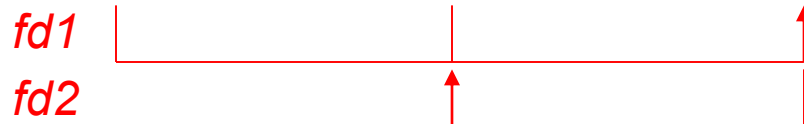


Figure 14.14
A relation TEACH that is in 3NF
but not BCNF.

Achieving the BCNF by Decomposition (1)

- Two FDs exist in the relation TEACH:
 - fd1: {student, course} \rightarrow instructor
 - fd2: instructor \rightarrow 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 meet this property, while possibly forgoing the preservation of all functional dependencies in the decomposed relations.
 - (See Algorithm 15.3)

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 **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 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) Property**
 - **Binary Decomposition:** Decomposition of a relation R into two relations.
 - **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 f.d. $((R_1 \cap R_2) \rightarrow (R_1 - R_2))$ is in F^+ , **or**
 - The f.d. $((R_1 \cap R_2) \rightarrow (R_2 - R_1))$ is in F^+ .

Closure of a set F of FDs is the set F^+ of all FDs 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** \rightarrow Instructor or **Student** \rightarrow Course, none of which is true.
- D2 gives **Course** \rightarrow Instructor or **Course** \rightarrow Student, none of which is true.
- However, in **D3** we get **Instructor** \rightarrow Course or **Instructor** \rightarrow Student.

Since **Instructor** \rightarrow Course is indeed true, the NJB property is satisfied and D3 is determined as a non-additive (good) decomposition.

General Procedure for achieving BCNF when a relation fails BCNF

Here we make use 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 $X \rightarrow A$ be the FD that causes a **violation of BCNF**. Then R may be decomposed into two relations:
- (i) $R - A$ and (ii) $X \cup A$.
- If either $R - A$ or $X \cup A$ 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 \cup Course), which gives the relations: (**Instructor**, **Student**) and (Instructor, Course) that we obtained before in decomposition D3.

5. Multivalued Dependency and Fourth Normal Form

- In many cases, relations have **constraints that cannot be specified as functional dependencies**. E.g., multivalued dependency.
- If we have **two or more multivalued *independent* attributes in the same relation**, we have to repeat every value of one of the attributes with every value of the other attribute to keep the relation consistent and to maintain the independence among the attributes involved.
 - This constraint is specified by a **multivalued dependency (MVD)**.
- Informally, **whenever two *independent* 1:N relationships A:B and A:C are mixed in the same relation, R(A, B, C), an MVD may arise**. e.g., PERSON(ssn, {email_address}, {phone#})

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

- (a) The EMP relation with two MVDs: $ENAME \twoheadrightarrow PNAME$ and $ENAME \twoheadrightarrow DNAME$.
- (b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

(a) **EMP**

<u>ENAME</u>	<u>PNAME</u>	<u>DNAME</u>
--------------	--------------	--------------

Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

(b) **EMP_PROJECTS**

<u>ENAME</u>	<u>PNAME</u>
--------------	--------------

Smith	X
Smith	Y

EMP_DEPENDENTS

<u>ENAME</u>	<u>DNAME</u>
--------------	--------------

Smith	John
Smith	Anna

5. Multivalued Dependencies and Fourth Normal Form (1)

Definition:

- A **multivalued dependency (MVD)** $X \twoheadrightarrow Y$ specified on relation schema R , where X and Y are both subsets of R , specifies the following constraint on any relation state r of R : If two tuples t_1 and t_2 exist in r such that $t_1[X] = t_2[X]$, then two tuples t_3 and t_4 should also exist in r with the following properties, where we use Z 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 Dependency and Fourth Normal Form (cont'd.)

- **Symmetry** of the definition
 - Whenever $X \twoheadrightarrow Y$ holds in R , so does $X \twoheadrightarrow Z$. Hence, $X \twoheadrightarrow Y$ implies $X \twoheadrightarrow Z$, and is written as $X \twoheadrightarrow Y \mid Z$
 - E.g. $ENAME \twoheadrightarrow PNAME$ and $ENAME \twoheadrightarrow DNAME$
(or $ENAME \twoheadrightarrow PNAME \mid DNAME$)
- A MVD $X \twoheadrightarrow Y$ in R is called **trivial MVD** if (a) Y is subset of X , **or** (b) $X \cup Y = R$.
 - An MVD that satisfies neither (a) nor (b) is called a **nontrivial MVD**.
 - If we have a nontrivial MVD in a relation, we may have to repeat values **redundantly**
 - Notice that relations containing nontrivial MVDs tend to be **all-key relations** – that is, their key is all their attributes taken together
 - An all-key relation is always in BCNF since it has no FDs

Multivalued Dependencies and Fourth Normal Form (3)

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 .
- The 4NF decomposition **not only can save on storage, but also avoid the update anomalies** associated with the MVDs.

Figure 14.15 Fourth and fifth normal forms.

(a) EMP

<u>Ename</u>	<u>Pname</u>	<u>Dname</u>
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

(c) SUPPLY

<u>Sname</u>	<u>Part_name</u>	<u>Proj_name</u>
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

(b) EMP_PROJECTS

<u>Ename</u>	<u>Pname</u>
Smith	X
Smith	Y

EMP_DEPENDENTS

<u>Ename</u>	<u>Dname</u>
Smith	John
Smith	Anna

(d) R_1

<u>Sname</u>	<u>Part_name</u>
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R_2

<u>Sname</u>	<u>Proj_name</u>
Smith	ProjX
Smith	ProjY
Adamsky	ProjY
Walton	ProjZ
Adamsky	ProjX

R_3

<u>Part_name</u>	<u>Proj_name</u>
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 \twoheadrightarrow Pname and Ename \twoheadrightarrow 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(R_1, R_2, R_3). (d) Decomposing the relation SUPPLY into the **5NF** relations R_1, R_2, R_3 .

6. Join Dependencies and Fifth Normal Form (1)

Definition:

- A **join dependency (JD)**, denoted by $JD(R_1, R_2, \dots, R_n)$, 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, \dots, R_n ; that is, for every such r we have
$$* (\pi_{R_1}(r), \pi_{R_2}(r), \dots, \pi_{R_n}(r)) = r$$

Note: an MVD is a special case of a JD where $n = 2$.

- A join dependency $JD(R_1, R_2, \dots, R_n)$, specified on relation schema R , is a **trivial JD** if one of the relation schemas R_i in $JD(R_1, R_2, \dots, R_n)$ is equal to R .

Join Dependencies and Fifth Normal Form (2)

Definition:

- A relation schema R is in **fifth normal form (5NF)** (or **Project-Join Normal Form (PJNF)**) with respect to a set F of functional, multivalued, and join dependencies if,
 - for every *nontrivial* join dependency $JD(R_1, R_2, \dots, R_n)$ 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.**

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

- Examples: additional constraint for Figure 14.15(c)
 - Whenever a supplier s supplies part p , and a project j uses part p , and the supplier s supplies at least one part to project j , then supplier s will also be supplying part p to project j .
 - The constraint can be restated and specifies a **join dependency** $JD(R1, R2, R3)$ among three projections $R1(Sname, Part_name)$, $R2(Sname, Proj_name)$, and $R3(Part_name, Proj_name)$
 - If the constraint holds, the tuples below the dashed line in Figure 14.15(c) must exist in any legal state of the SUPPLY relation that also contains the tuples above the dashed line
 - Notice that applying a natural join to *any two of these relations* produces **spurious tuples**, but applying a natural join to *all three together* does not

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)
- Fourth and Fifth Normal Forms