

Database Design Theory and Normalization

-Navathe, 6th edition

15.1 Informal Design Guidelines for Relation Schemas

- Making sure that the semantics of the attributes is clear in the schema
- Reducing the redundant information in tuples
- Reducing the NULL values in tuples
- Disallowing the possibility of generating spurious tuples

15.1.1 Imparting Clear Semantics to Attributes in Relations

- **Guideline 1**
- Design a 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 interpret and to explain its meaning. Otherwise, if the relation corresponds to a mixture of multiple entities and relationships, semantic ambiguities will result and the relation cannot be easily explained.

15.1.1 Imparting Clear Semantics to Attributes in Relations

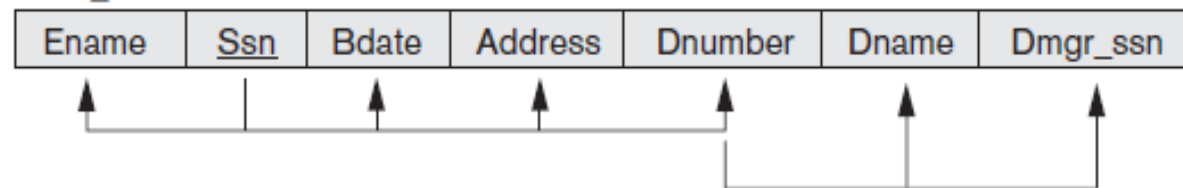
- Below example shows violating Guideline 1 by mixing attributes from distinct real-world entities: EMP_DEPT mixes attributes of employees and departments, and EMP_PROJ mixes attributes of employees and projects and the WORKS_ON relationship.

Figure 15.3

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

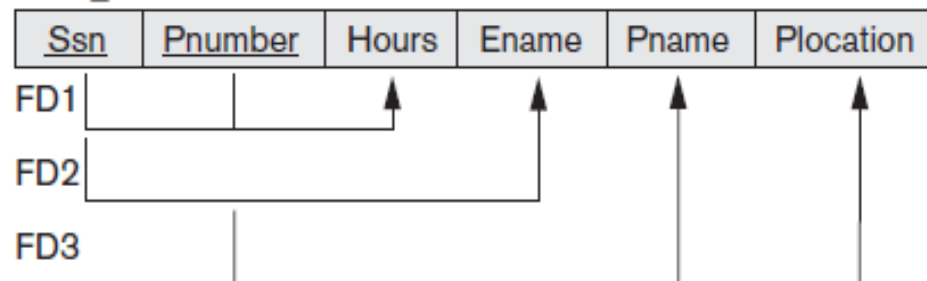
(a)

EMP_DEPT



(b)

EMP_PROJ



15.1.2 Redundant Information in Tuples and Update Anomalies

- **Guideline 2**
- Design the base relation schemas so that no insertion, deletion, or modification anomalies are present in the relations. If any anomalies are present, note them clearly and make sure that the programs that update the database will operate correctly.

15.1.2 Redundant Information in Tuples and Update Anomalies

- Update Anomalies: Insertion, Deletion and Modification anomalies occur because of poor relation schema design.
- Suppose instead of using the correct relation schema given below
 - EMPLOYEE(ssn, ename, bdate, address, dno)
 - DEPARTMENT(dname, dnumber, dmgr_ssn)
 - DEPT_LOCATIONS(dnum, dlocation)
 - WORKS_ON(ssn, pnumber, hours)
 - PROJECT(pnumber, pname, plocation, dnum)
- A bad relation schema is used like given below we can see the anomalies:
 - EMP_DEPT(ssn, ename, bdate, address, dnumber, dname, dmgr_ssn)
 - EMP_PROJECT(ssn, ename, pnumber, pname, plocation, hours)

Redundant information

EMPLOYEE		
<u>Employee-id</u>	Name	Dnumber
110	Jim	2
111	Kim	5
112	Tim	5

DEPARTMENT		
<u>Dnumber</u>	Dname	Manager
2	Finance	Kim
5	Account	Jim

No redundant information

EMPLOYEE-DEPARTMENT				
<u>Employee-id</u>	Name	Dnumber	Dname	Manager
110	Jim	2	Finance	Kim
111	Kim	5	Account	Jim
112	Tim	5	Account	Jim


Redundant information

15.1.2 Redundant Information in Tuples and Update Anomalies

Insertion anomaly:

- Suppose in EMP_DEPT table a new department has to be added which does not yet contain any employees, it is not possible as we have to enter the values for the employee related tuple. Nulls cannot be entered as ssn is the primary key of the table.
- Similarly if we want to enter details of new employee, all the details related to the department has to be repeated redundantly.

Insertion Anomaly

 EMPLOYEE-DEPARTMENT

<u>Employee_id</u>	Name	Dnumber	Dname	Manager
110	Jim	2	Finance	Kim
111	Kim	5	Account	Jim
112	Tim	5	Account	Jim
NULL	NULL	10	Sales	James

15.1.2 Redundant Information in Tuples and Update Anomalies

Deletion anomaly:

- Suppose we want to delete the last employee who works in a particular department, doing so will delete all the details of that department completely

Deletion Anomaly

EMPLOYEE-DEPARTMENT				
<u>Employee_id</u>	Name	Dnumber	Dname	Manager
110	Jim	2	Finance	Kim
111	Kim	5	Account	Jim
112	Tim	5	Account	Jim
115	Red	7	Sales	Blue

15.1.2 Redundant Information in Tuples and Update Anomalies

- Modification anomalies
- Suppose we want to change the name of the manager for a particular department, the modification should be done on all the tuples having that value.

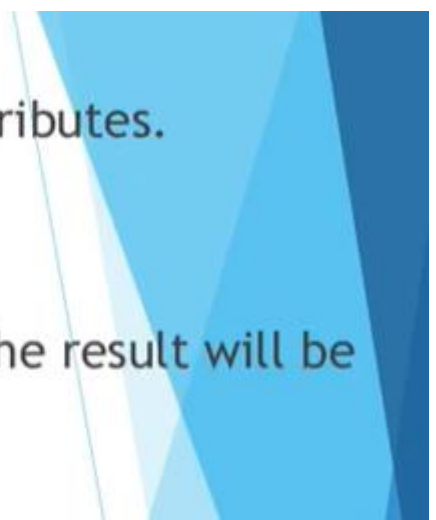
Modification anomaly

EMPLOYEE-DEPARTMENT				
<u>Employee_id</u>	Name	Dnumber	Dname	Manager
110	Jim	2	Finance	Kim
111	Kim	5	Account	Jim
112	Tim	5	Account	Jim
115	Red	7	Sales	Blue

15.1.3 NULL Values in Tuples

- **Guideline 3**
- As far as possible, 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 and do not apply to a majority of tuples in the relation.
- For example, if only 15 percent of employees have individual offices, there is little justification for including an attribute `Office_number` in the `EMPLOYEE` relation; rather, a relation `EMP_OFFICES(Essn, Office_number)` can be created to include tuples for only the employees with individual offices.

15.1.3 NULL Values in Tuples

- ▶ Null values lead to problems with understanding the meaning of the attributes.
 - ▶ Also it causes confusions in COUNT and SUM operations.
 - ▶ Also if NULL value comes in comparison in SELECT or JOIN operations, the result will be unpredictable.
- 

15.1.4 Generation of Spurious Tuples

- **Guideline 4**
- Design relation schemas so that they can be joined with equality conditions on attributes that are appropriately related (primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated. Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations because joining on such attributes may produce spurious tuples.

15.1.4 Generation of Spurious Tuples

Emp_prj
name|plocation

a|blore
b|hyd
c|delhi
d|mumbai

prj
pno |ssn | plocation

1|11|blore
2|22|blore
3|33|hyd
4|44|mumbai
5|55|mumbai

name|plocat pno |ssn | plocation

a|blore|1|11|blore
a|blore|2|22|blore
b|hyd|3|33|hyd
d|mumbai|4|44|mumbai
d|mumbai|5|55|mumbai

Generation of spurious tuples as a result of join on plocation attribute which is not a primary key

ssn name|plocat pno |ssn | plocation

11|a|blore|1|11|blore
22|b|blore|2|22|blore
33|c|delhi|3|33|delhi
44|d|mumbai|4|44|mumbai

Correct result if ssn was in the emp_prj table and join was on that attribute.

15.2 Functional Dependencies

- A functional dependency is a constraint between two sets of attributes from the database
- Suppose we have a **universal** relation schema $R = \{A_1, A_2, \dots, A_n\}$. 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 $t_1[Y] = t_2[Y]$.
- The values of the X component of a tuple uniquely (or **functionally**) *determine* the values of the Y component.
- There is a functional dependency from X to Y , Y is **functionally dependent** on X
- Thus, X functionally determines Y in a relation schema R if, and only if, whenever two tuples of $r(R)$ agree on their X -value, they must necessarily agree on their Y value.

Valid: Y is Determined by X

Sid	Sname
1	Ranjit
2	Varun

Valid:

Sid	Sname
1	Ranjit
1	Ranjit

~~Invalid~~:

Sid	Sname
1	Ranjit
1	Varun

~~Invalid~~:

Sid	Sname
1	Ranjit
2	Ranjit

Reflexivity: if Y is subset of X then $X \rightarrow Y$

Augmentation: if $X \rightarrow Y$, then $XZ \rightarrow YZ$

Transitive: if $X \rightarrow Y$ and $Y \rightarrow Z$ then $X \rightarrow Z$

Union: if $X \rightarrow Y$ and $X \rightarrow Z$ then $X \rightarrow YZ$

Decomposition: if $X \rightarrow YZ$ then $X \rightarrow Y$ and $X \rightarrow Z$

Pseudotransitivity: if $X \rightarrow Y$ and $WY \rightarrow Z$ then $WX \rightarrow Z$

Composition: if $X \rightarrow Y$ and $Z \rightarrow W$ then $XZ \rightarrow YW$

15.3.1 Normalization of Relations

- **Normalization of data** can be considered a process of analyzing the given relation schemas based on their FDs and primary keys to achieve the desirable properties of (1) minimizing redundancy and (2) minimizing the insertion, deletion, and update anomalies.
- 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 tests that can be carried out on individual relation schemas so that the relational database can be **normalized** to any desired degree

15.3.1 Normalization of Relations

- 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.
- The process of normalization through decomposition must also confirm the existence of two additional properties:
 - The **nonadditive join or lossless join property**, which guarantees that the spurious tuple generation problem does not occur with respect to the relation schemas created after decomposition.
 - The **dependency preservation property**, which ensures that each functional dependency is represented in some individual relation resulting after decomposition.

15.3.2 Practical Use of Normal Forms

- Database design as practiced in industry today pay particular attention to normalization only up to 3NF, BCNF, or at most 4NF.
- Designers *need not* normalize to the highest possible normal form. Relations may be left in a lower normalization status, such as 2NF, for performance reasons.
- **Definition. Denormalization** is the process of storing the join of higher normal form relations as a base relation, which is in a lower normal form.

15.3.3 Definitions of Keys and Attributes Participating in Keys

- An attribute of relation schema R is called a **prime attribute** of R if it is a member of *some candidate key* of R .
- An attribute is called **nonprime** if it is not a prime attribute—that is, if it is not a member of any candidate key.
- For ex:- Both Ssn and Pnumber are prime attributes of WORKS_ON, whereas other attributes of WORKS_ON are nonprime.

15.3.4 First Normal Form

- **First normal form (1NF)** states that the domain of an attribute must include only *atomic* (simple, indivisible) *values* and that the value of any attribute in a tuple must be a *single value* from the domain of that attribute.
- Hence, 1NF disallows having a set of values, a tuple of values, or a combination of both as an attribute value for a *single tuple*.
- In other words, 1NF disallows *relations within relations* or *relations as attribute values within tuples*. The only attribute values permitted by 1NF are single **atomic** (or **indivisible**) **values**.

15.3.4 First Normal Form

- Example of relation not in 1NF form

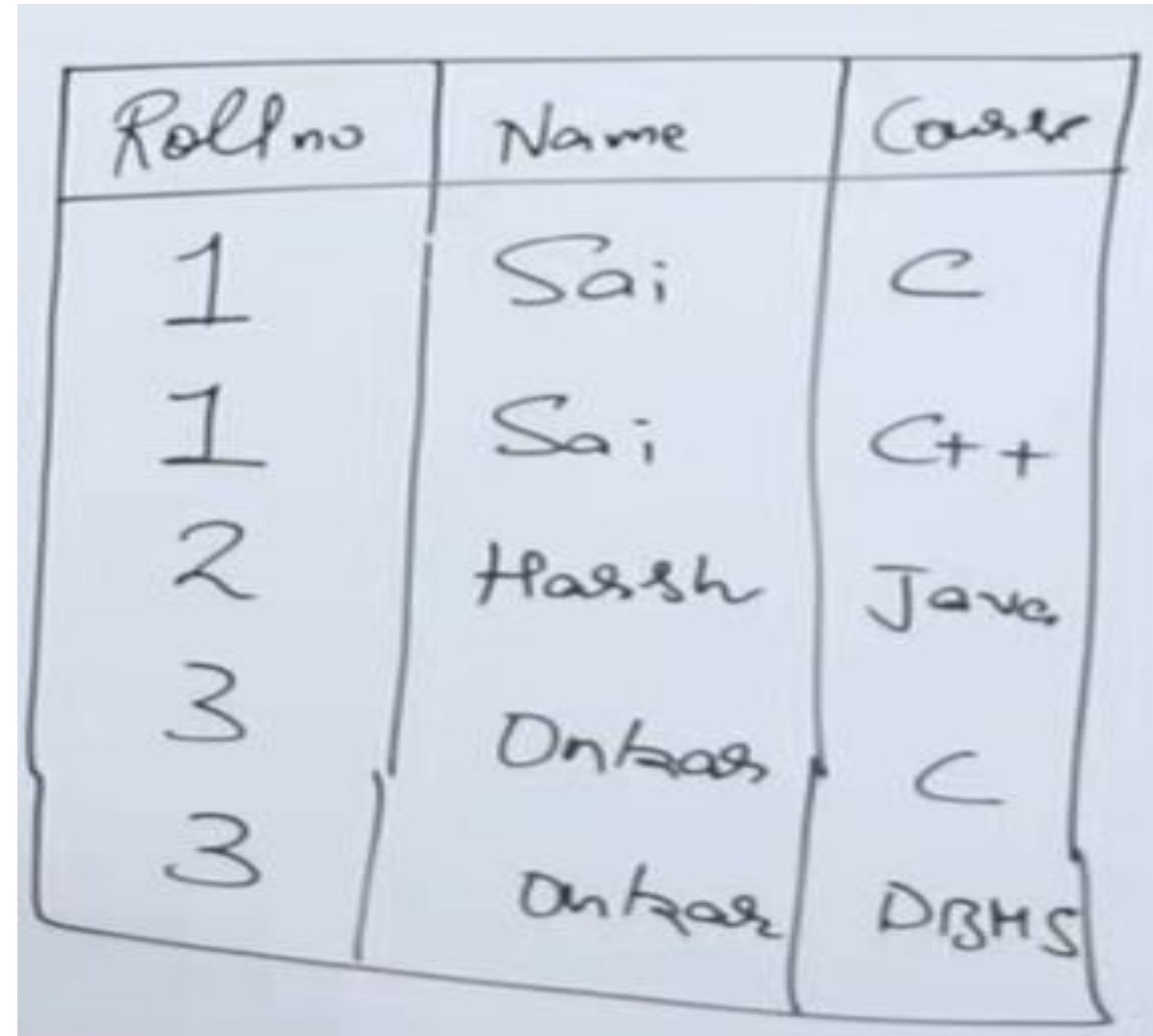
Rollno	Name	Course
1	Sai	C/C++
2	Harsh	Java
3	Onkar	C/DBMS

15.3.4 First Normal Form

- There are three main techniques to achieve first normal form for such a relation:
- **1.** Remove the attribute that violates 1NF and place it in a separate relation. This decomposes the non-1NF relation into two 1NF relations.
- **2.** Expand the key so that there will be a separate tuple in the original relation for each value of the attribute. In this case, the primary key becomes the combination of earlier primary key and the multivalued attribute. This solution has the disadvantage of introducing *redundancy* in the relation.
- **3.** If a *maximum number of values* is known for the attribute, replace the multi-valued attribute by its atomic attribute values. This solution has the disadvantage of introducing *NULL values*.

15.3.4 First Normal Form

- 2. Expand the key so that there will be a separate tuple in the original relation for each value of the attribute. In this case, the primary key becomes the combination of earlier primary key and the multivalued attribute. This solution has the disadvantage of introducing *redundancy* in the relation.



Rollno	Name	Course
1	Sai	C
1	Sai	C++
2	Harsh	Java
3	Ontkar	C
3	Ontkar	DBMS

Rollno and Course together will be the primary key

15.3.4 First Normal Form

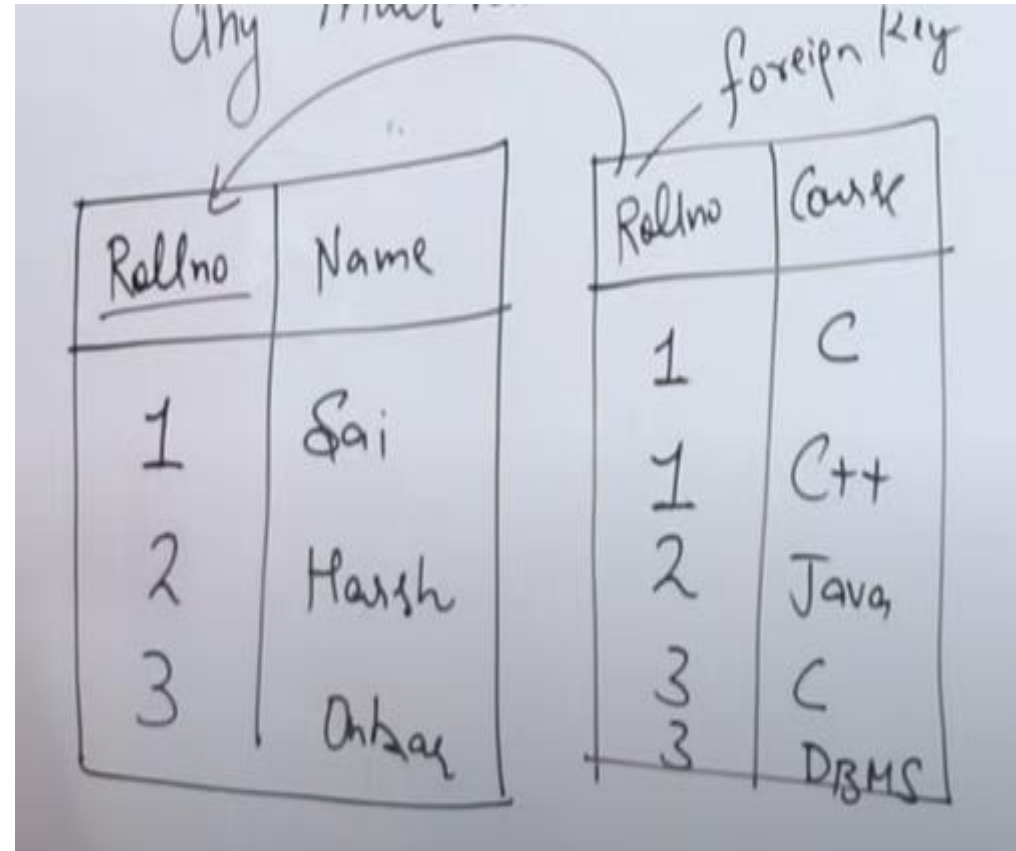
- 3. If a *maximum number of values* is known for the attribute, replace the multi-valued attribute by its atomic attribute values. This solution has the disadvantage of introducing *NULL values*.

Primary Key: Rollno

<u>Rollno</u>	Name	Course1	Course2
1	Sai	C	C++	- - -
2	Harsh	<u>Java</u>	<u>Null</u>	Null Null
3	Onkar	C	DBMS	

15.3.4 First Normal Form

- 1. Remove the attribute that violates 1NF and place it in a separate relation. This decomposes the non-1NF relation into two 1NF relations.

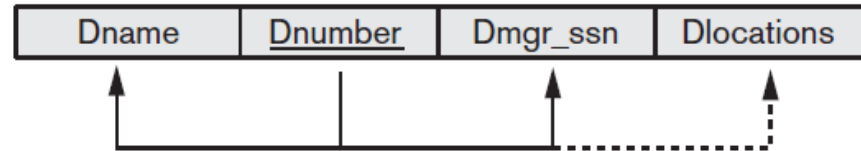


- This is the most feasible solution

15.3.4 First Normal Form

(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 15.9

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.

15.3.4 First Normal Form

Another example:

PERSON (Ss#, {Car_lic#}, {Phone#})

Decomposed into

P1(Ss#, Car_lic#) and

P2(Ss#, Phone#).

Figure 15.10

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

(a)

EMP_PROJ		Projs	
Ssn	Ename	Pnumber	Hours

(b)

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)

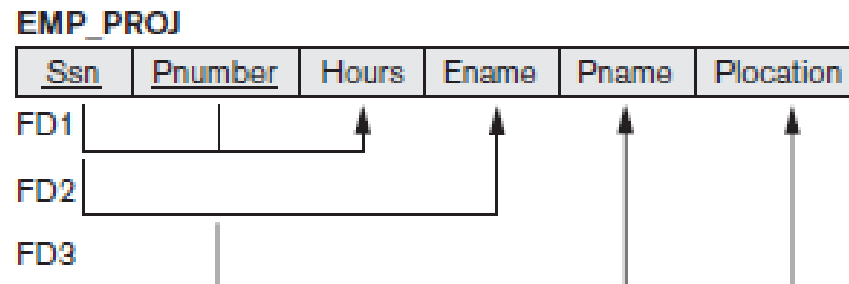
EMP_PROJ1	
Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours
-----	---------	-------

15.3.5 Second Normal Form

- **Second normal form (2NF)** is based on the concept of *full functional dependency*. A functional dependency $X \rightarrow Y$ is a **full functional dependency** if removal of any attribute A from X means that the dependency does not hold any more; that is, for any attribute $A \in X$, $(X - \{A\})$ 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; that is, for some $A \in X$, $(X - \{A\}) \rightarrow Y$.
- In Figure 15.3(b), $\{Ssn, Pnumber\} \rightarrow Hours$ is a full dependency (neither $Ssn \rightarrow Hours$ nor $Pnumber \rightarrow Hours$ holds).
- However, the dependency $\{Ssn, Pnumber\} \rightarrow Ename$ is partial because $Ssn \rightarrow Ename$ holds.



15.3.5 Second Normal Form

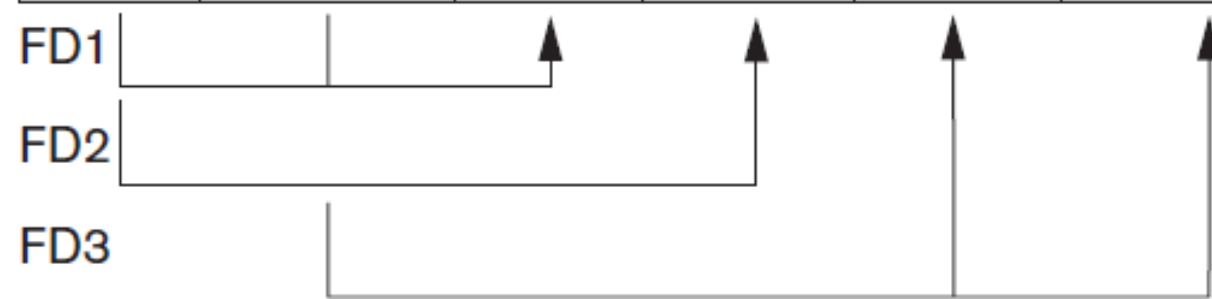
- **Definition:** A relation schema R is in 2NF if every nonprime attribute A in R is *fully functionally dependent* on the primary key of R .
- The test for 2NF involves testing for functional dependencies whose left-hand side attributes are part of the primary key.
- If the primary key contains a single attribute, the test need not be applied at all.
- The EMP_PROJ relation in Figure is in 1NF but is not in 2NF. The nonprime attribute Ename violates 2NF because of FD2, as do the nonprime attributes Pname and Plocation because of FD3.
- The functional dependencies FD2 and FD3 make Ename, Pname, and Plocation partially dependent on the primary key {Ssn, Pnumber} of EMP_PROJ, thus violating the 2NF test.
- To convert it to it is decomposed into the three relation schemas EP1, EP2, and EP3 shown in Figure 15.11(a), each of which is in 2NF.

15.3.5 Second Normal Form

(a)

EMP_PROJ

<u>Ssn</u>	<u>Pnumber</u>	Hours	Ename	Pname	Plocation
------------	----------------	-------	-------	-------	-----------



2NF Normalization

EP1

<u>Ssn</u>	<u>Pnumber</u>	Hours
------------	----------------	-------



EP2

<u>Ssn</u>	Ename
------------	-------



EP3

<u>Pnumber</u>	Pname	Plocation
----------------	-------	-----------



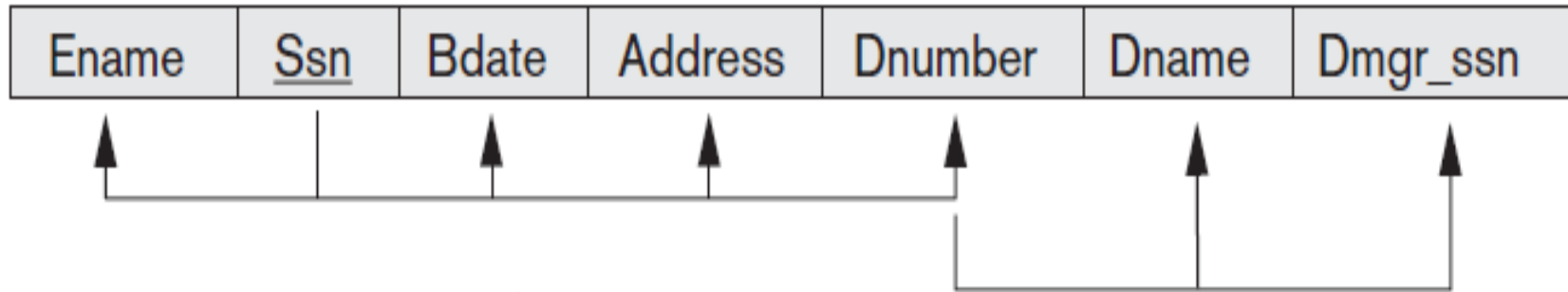
15.3.6 Third Normal Form

- **Third normal form (3NF)** is based on the concept of *transitive dependency*. A functional dependency $X \rightarrow Y$ in a relation schema R is a **transitive dependency** if there exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R , and both $X \rightarrow Z$ and $Z \rightarrow Y$ hold.
- The dependency $Ssn \rightarrow Dmgr_ssn$ is transitive through $Dnumber$ in EMP_DEPT in Figure, because both the dependencies $Ssn \rightarrow Dnumber$ and $Dnumber \rightarrow Dmgr_ssn$ hold *and* $Dnumber$ is neither a key itself nor a subset of the key of EMP_DEPT .
- Intuitively, we can see that the dependency of $Dmgr_ssn$ on $Dnumber$ is undesirable in EMP_DEPT since $Dnumber$ is not a key of EMP_DEPT .
- **Definition.** According to Codd's original definition, a relation schema R is in **3NF** if it satisfies 2NF *and* no nonprime attribute of R is transitively dependent on the primary key.

15.3.6 Third Normal Form

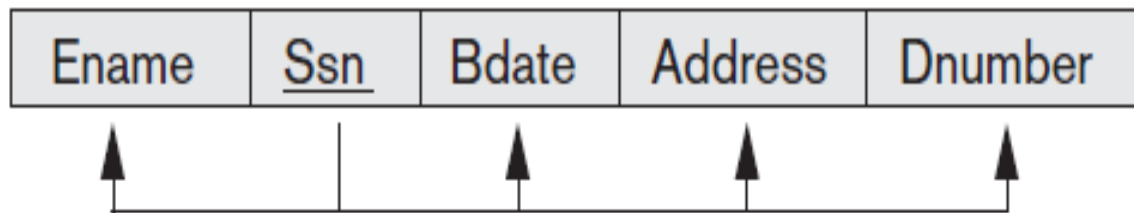
(b)

EMP_DEPT

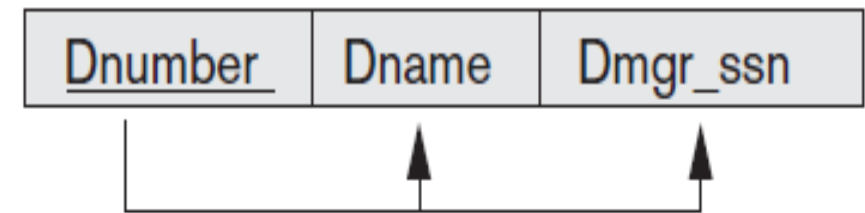


3NF Normalization

ED1



ED2



15.4 General Definitions of Second and Third Normal Forms

Table 15.1 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).

15.4.1 General Definition of Second Normal Form

- **Definition.** A relation schema R is in **second normal form (2NF)** if every nonprime attribute A in R is not partially dependent on *any* key of R .
- Consider the relation schema LOTS shown in Figure 15.12(a), which describes parcels of land for sale in various counties of a state.
- Suppose that there are two candidate keys: Property_id\# and $\{\text{County_name}, \text{Lot\#}\}$; that is, lot numbers are unique only within each county, but Property_id\# numbers are unique across counties for the entire state.
- Based on the two candidate keys Property_id\# and $\{\text{County_name}, \text{Lot\#}\}$, the functional
- dependencies FD1 and FD2 in Figure 15.12(a) hold.
- We choose Property_id\# as the primary key, so it is underlined in Figure 15.12(a), but no special consideration will be given to this key over the other candidate key.

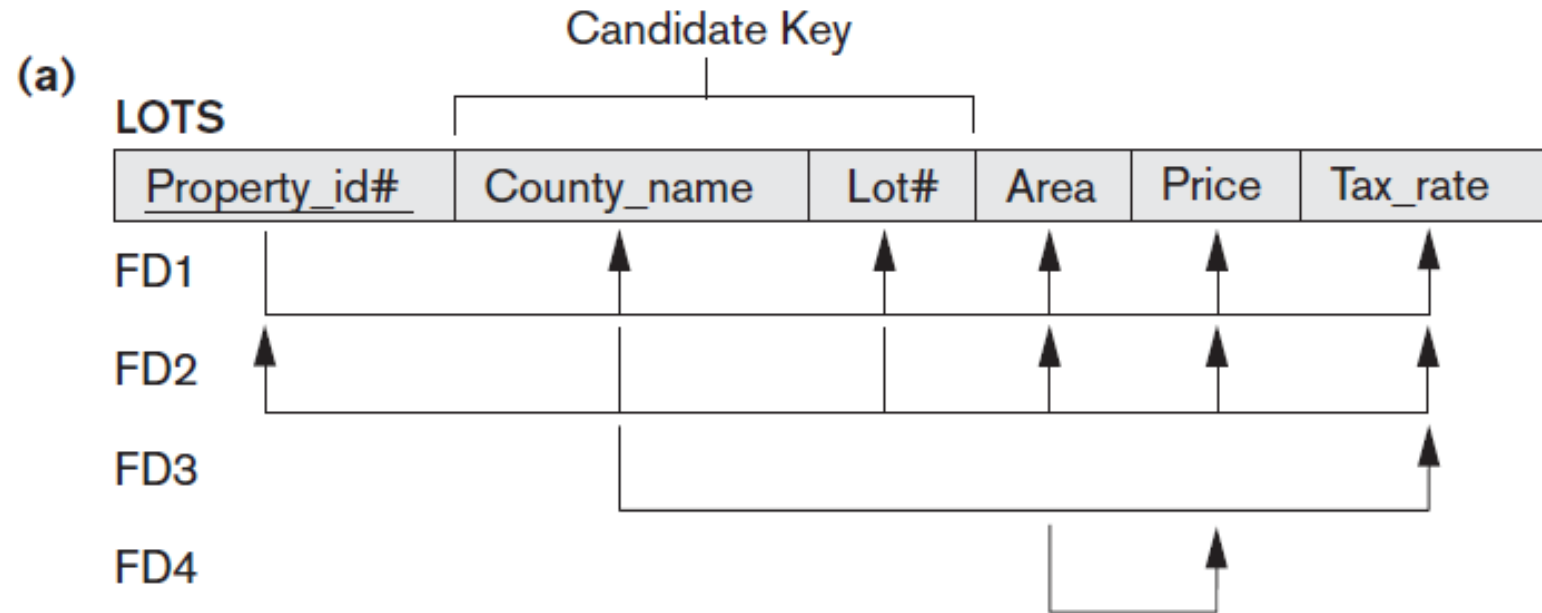
15.4.1 General Definition of Second Normal Form

- Suppose that the following two additional functional dependencies hold in LOTS:
- FD3: $\text{County_name} \rightarrow \text{Tax_rate}$
- FD4: $\text{Area} \rightarrow \text{Price}$
- In words, the dependency FD3 says that the tax rate is fixed for a given, while FD4 says that the price of a lot is determined by its area regardless of which county it is in. The LOTS relation schema violates the general definition of 2NF because Tax_rate is partially dependent on the candidate key $\{\text{County_name}, \text{Lot\#}\}$, due to FD3.
- To normalize LOTS into 2NF, we decompose it into the two relations LOTS1 and LOTS2, shown in Figure 15.12(b).
- We construct LOTS1 by removing the attribute Tax_rate that violates 2NF from LOTS and placing it with County_name into another relation LOTS2. Both LOTS1 and LOTS2 are in 2NF. Notice that FD4 does not violate 2NF and is carried over to LOTS1.

15.4.1 General Definition of Second Normal Form

Figure 15.12

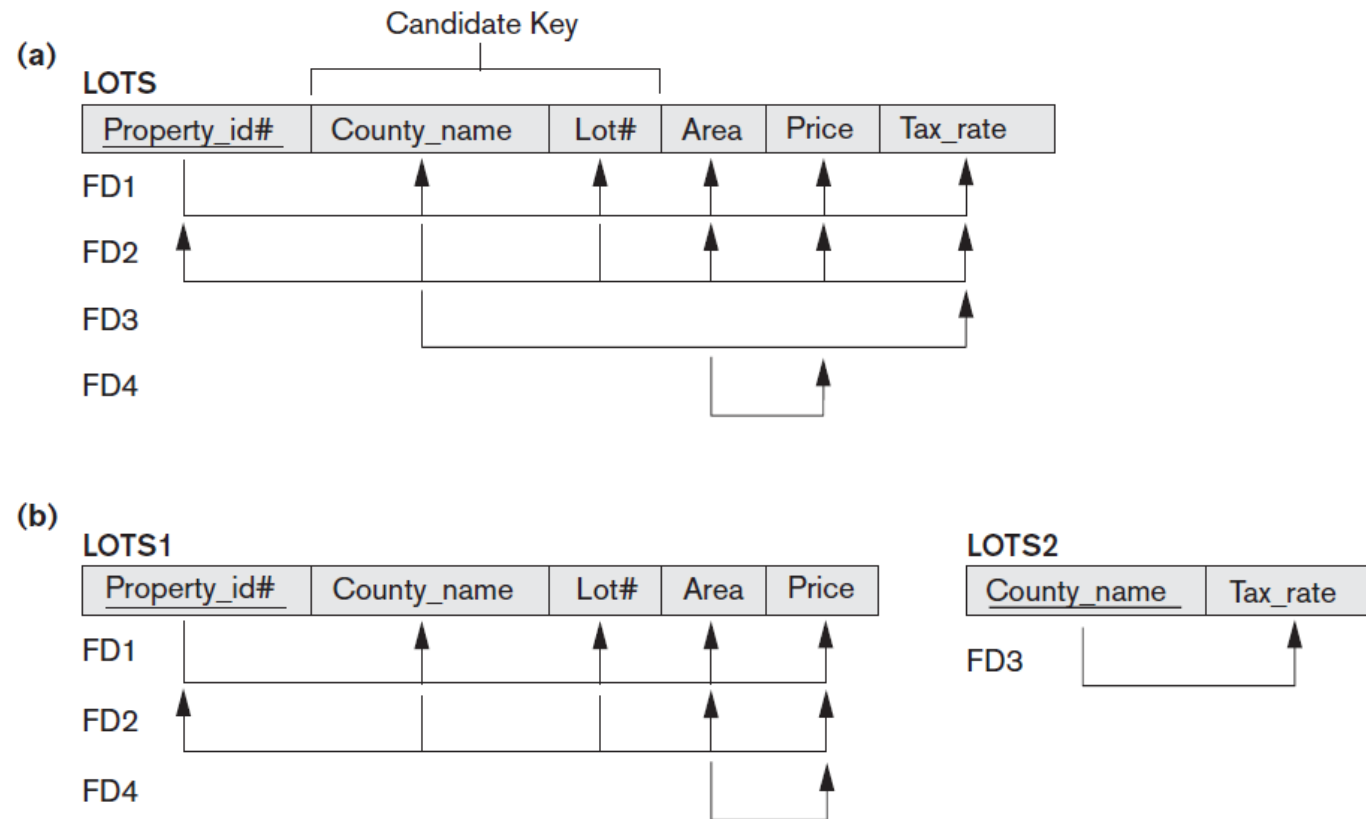
Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of the progressive normalization of LOTS.



15.4.1 General Definition of Second Normal Form

Figure 15.12

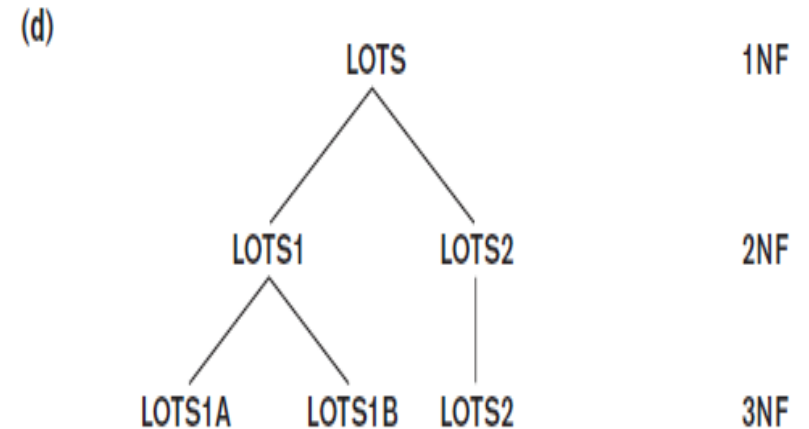
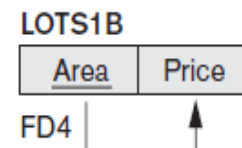
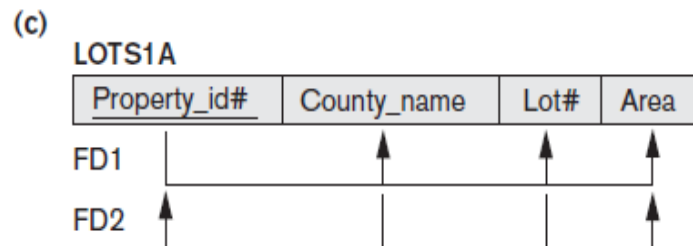
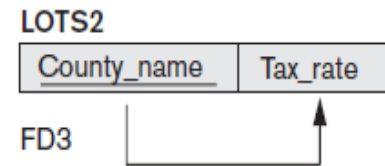
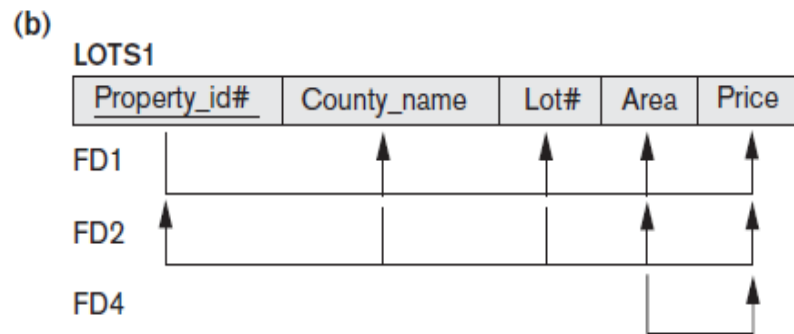
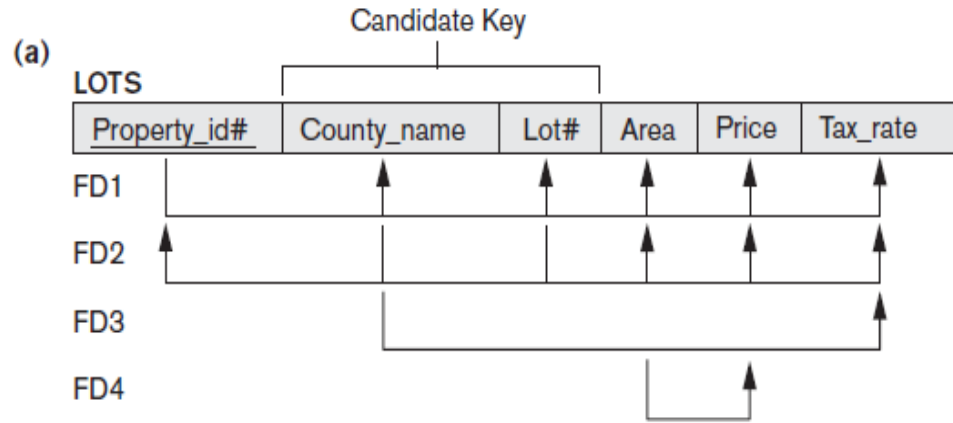
Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of the progressive normalization of LOTS.



15.4.3 Interpreting the General Definition of Third Normal Form

- A relation schema R violates the general definition of 3NF if a functional dependency $X \rightarrow A$ holds in R that does not meet either condition—meaning that it violates *both* conditions (a) and (b) of 3NF. This can occur due to two types of problematic functional dependencies:
 - ■ A nonprime attribute determines another nonprime attribute. Here we typically have a transitive dependency that violates 3NF.
 - ■ A proper subset of a key of R functionally determines a nonprime attribute.
- Here we have a partial dependency that violates 3NF (and also 2NF). Therefore, we can state a **general alternative definition of 3NF** as follows:
- **Alternative Definition.** A relation schema R is in 3NF if every nonprime attribute of R meets both of the following conditions:
 - ■ It is fully functionally dependent on every key of R .
 - ■ It is nontransitively dependent on every key of R .

15.4.3 Interpreting the General Definition of Third Normal Form



15.5 Boyce-Codd Normal Form

- Suppose that we have thousands of lots in the relation but the lots are from only two counties: DeKalb and Fulton.
- Suppose also that lot sizes in DeKalb County are only 0.5, 0.6, 0.7, 0.8, 0.9, and 1.0 acres, whereas lot sizes in Fulton County are restricted to 1.1, 1.2, ..., 1.9, and 2.0 acres.
- In such a situation we would have the additional functional dependency FD5: $\text{Area} \rightarrow \text{County_name}$.
- If we add this to the other dependencies, the relation schema LOTS1A still is in 3NF because County_name is a prime attribute.
- If we add this to the other dependencies, the relation schema LOTS1A still is in 3NF because County_name is a prime attribute.
- BCNF is a *stronger normal form* that would disallow LOTS1A and suggest the need for decomposing it.

15.5 Boyce-Codd Normal Form

- **Definition.** A relation schema R is in **BCNF** if whenever a *nontrivial* functional dependency $X \rightarrow A$ holds in R , then X is a superkey of R .
- In the example, FD5 violates BCNF in LOTS1A because AREA is not a superkey of LOTS1A.
- We can decompose LOTS1A into two BCNF relations LOTS1AX and LOTS1AY, shown in Figure 15.13(a).
- This decomposition loses the functional dependency FD2 because its attributes no longer coexist in the same relation after decomposition.

15.5 Boyce-Codd Normal Form

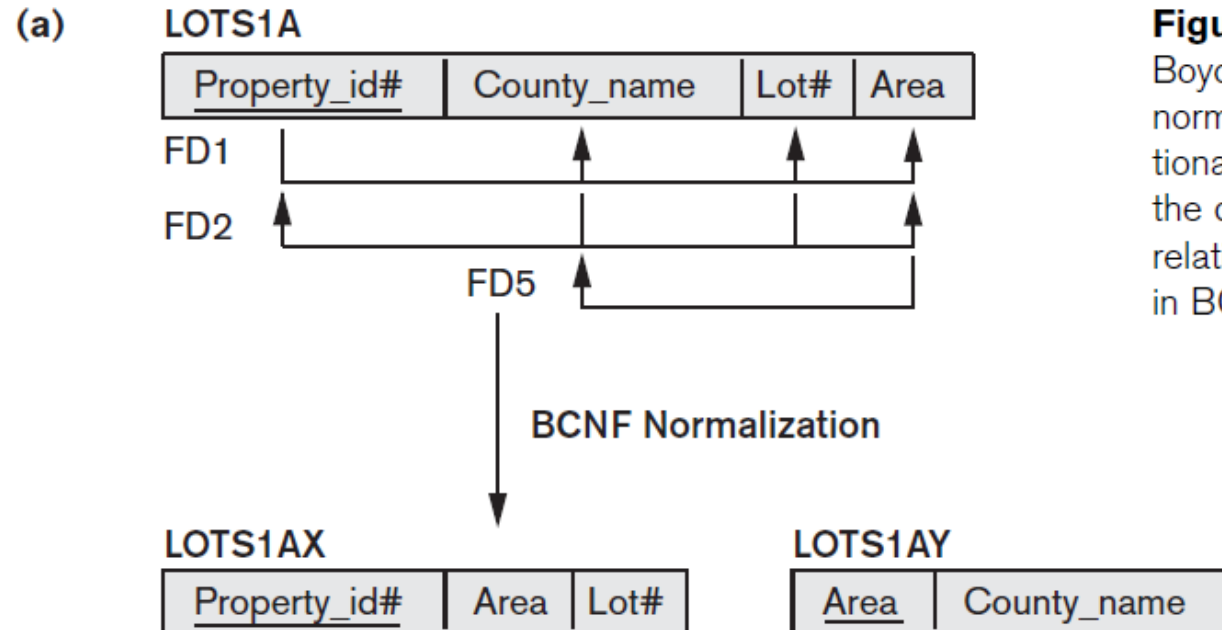
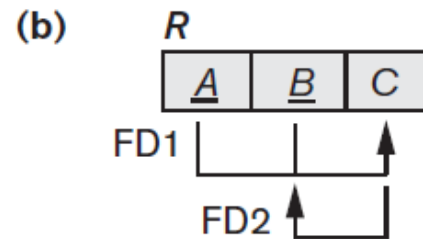


Figure 15.13

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.



As another example, consider Figure 15.14, which shows a relation TEACH with the following dependencies:

FD1: {Student, Course} → Instructor

FD2: Instructor → Course

15.5 Boyce-Codd Normal Form

- This relation is in 3NF but not BCNF. Decomposition of this relation schema into two schemas is not straightforward because it may be decomposed into one of the three following possible pairs:
 1. {Student, Instructor} and {Student, Course}.
 2. {Course, Instructor} and {Course, Student}.
 3. {Instructor, Course} and {Instructor, Student}.
- All three decompositions *lose* the functional dependency FD1.
- The *desirable decomposition* of those just shown is 3 because it will not generate spurious tuples after a join.
- Algorithm 16.5 does that and could be used above to give decomposition 3 for TEACH, which yields two relations in BCNF as:
 - (Instructor, Course) and (Instructor, Student)

15.6 Multivalued Dependency and Fourth Normal Form

- Many cases relations have constraints that cannot be specified as functional dependencies.
- *Fourth normal form*, is based on multivalued dependency.
- If we have two or more multivalued *independent* attributes in the same relation schema, we get into a problem of having to repeat every value of one of the attributes with every value of the other attribute to keep the relation state consistent and to maintain the independence among the attributes involved.
- This constraint is specified by a multivalued dependency.
- For example, consider the relation EMP shown in Figure 15.15(a). A tuple in this EMP relation represents the fact that an employee whose name is Ename works on the project whose name is Pname and has a dependent whose name is Dname.
- An employee may work on several projects and may have several dependents, and the employee's projects and dependents are independent of one another.

15.6 Multivalued Dependency and Fourth Normal Form

- To keep the relation state consistent, and to avoid any spurious relationship between the two independent attributes, we must have a separate tuple to represent every combination of an employee's dependent and an employee's project.
- 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.

(a) EMP

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

15.6.1 Formal Definition of Multivalued Dependency Definition.

- A multivalued dependency $X \twoheadrightarrow Y$ specified on relation schema R , where X and Y are both subsets of R , specifies the following constraint on any relation state r of R : If two tuples t_1 and t_2 exist in r such that $t_1[X] = t_2[X]$, 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 $t_4[Z] = t_1[Z]$.
- Whenever $X \twoheadrightarrow Y$ holds, we say that X **multidetermines** Y . Because of the symmetry in the definition, whenever $X \twoheadrightarrow Y$ holds in R , so does $X \twoheadrightarrow Z$. Hence, $X \twoheadrightarrow Y$ implies $X \twoheadrightarrow Z$, and therefore it is sometimes written as $X \twoheadrightarrow Y|Z$.

15.6.1 Formal Definition of Multivalued Dependency Definition.

- (a) The EMP relation with two MVDs: $\text{Ename} \twoheadrightarrow \text{Pname}$ and $\text{Ename} \twoheadrightarrow \text{Dname}$.
- (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

15.6.1 Formal Definition of Multivalued Dependency Definition.

- An MVD $X \twoheadrightarrow Y$ in R is called a **trivial MVD** if (a) Y is a subset of X , or (b) $X \cup Y = R$. For example, the relation EMP_PROJECTS in Figure 15.15(b) has the trivial
- MVD $Ename \twoheadrightarrow Pname$. An MVD that satisfies neither (a) nor (b) is called a **nontrivial MVD**.
- A trivial MVD will hold in *any* relation state r of R ; it is called trivial because it does not specify any significant or meaningful constraint on R .
- If we have a *nontrivial MVD* in a relation, we may have to repeat values redundantly in the tuples.

(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

15.6.1 Formal Definition of Multivalued Dependency Definition.

- **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 .
- We can state the following points:
 - An all-key relation is always in BCNF since it has no FDs.
 - An all-key relation such as the EMP relation in Figure 15.15(a), which has no FDs but has the MVD $\text{Ename} \twoheadrightarrow \text{Pname} \mid \text{Dname}$, is not in 4NF.
 - A relation that is not in 4NF due to a nontrivial MVD must be decomposed to convert it into a set of relations in 4NF.
 - The decomposition removes the redundancy caused by the MVD.

15.7 Join Dependencies and Fifth Normal Form

- Another dependency called the *join dependency* may be present, in that case a *multiway decomposition* into fifth normal form (5NF) will be done.
- It is important to note that such a dependency is a very peculiar semantic constraint that is very difficult to detect in practice; therefore, normalization into 5NF is very rarely done in practice.
- **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 nonadditive join decomposition into R_1, R_2, \dots, R_n . Hence, for every such r we have $* (\pi_{R_1}(r), \pi_{R_2}(r), \dots, \pi_{R_n}(r)) = r$
- 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 .
- Such a dependency is called trivial because it has the nonadditive join property for any relation state r of R and thus does not specify any constraint on R .
- Fifth normal form, is also called *project-join normal form*.

15.7 Join Dependencies and Fifth Normal Form

- **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 .
- For an example of a JD, consider once again the SUPPLY all-key relation in Figure 15.15(c).
- Suppose that the following additional constraint always holds: 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 .
- This constraint can be restated in other ways and specifies a join dependency $JD(R_1, R_2, R_3)$ among the three projections $R_1(\text{Sname}, \text{Part_name})$, $R_2(\text{Sname}, \text{Proj_name})$, and $R_3(\text{Part_name}, \text{Proj_name})$ of SUPPLY.

15.7 Join Dependencies and Fifth Normal Form

- Figure 15.15(d) shows how the SUPPLY relation *with the join dependency* is decomposed into three relations R_1 , R_2 , and R_3 that are each in 5NF.
- 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.

15.7 Join Dependencies and Fifth Normal Form

(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 .

(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

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

END of chapter 15