
CS3402 : Chapter 4

Normal Forms

Functional Dependency

- Functional dependency is a constraint between two sets of attributes from the database
- Formal definition:
 - ◆ Let R be a relation schema, and $\alpha \subseteq R$, $\beta \subseteq R$ (i.e., α and β are sets of R 's attributes). We say:

$$\alpha \rightarrow \beta$$

- ◆ If in any relation instance $r(R)$, for all pairs of tuples $t1$ and $t2$ in r , we have:

$$(t1[\alpha] = t2[\alpha]) \Rightarrow (t1[\beta] = t2[\beta])$$

Inference Rules for FDs

IR1 (reflective rule)	If X is a subset of Y , then $X \rightarrow Y$
IR2 (augmentation rule)	If $X \rightarrow Y$, then $XZ \rightarrow YZ$
IR3 (transitive rule)	If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
IR4 (decomposition rule)	If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
IR5 (union rule)	If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
IR6 (pseudotransitive rule)	If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$

Inference Rules for FDs

- **Closure** of a set of **attributes** X with respect to F is the set X^+ of all attributes that are functionally determined by X
 - ◆ Note both X and X^+ are a set of attributes
- If X^+ consists of all attributes of R , X is a **superkey** for R
 - ◆ From the value of X , we can determine the values the whole tuple
- X^+ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F
- From X to find out X^+

Inference Rules for FDs

- **Closure** of a set F of **FDs** is the set F^+ of **all FDs** that can be inferred from F .
- Two sets of FDs F and G are **equivalent** if:
 - ◆ Every FD in F can be inferred from G , and
 - ◆ Every FD in G can be inferred from F
 - ◆ Hence, F and G are equivalent if $F^+ = G^+$

Relational Database Design

- Logical / Conceptual DB Design
 - ◆ Schema
 - ◆ what relations (tables) are needed?
 - ◆ what their attributes should be?
- What is a “bad” DB Design?
 - Repetition of data/information
 - Potential inconsistency
 - Inability to represent certain information
 - Loss of data/information

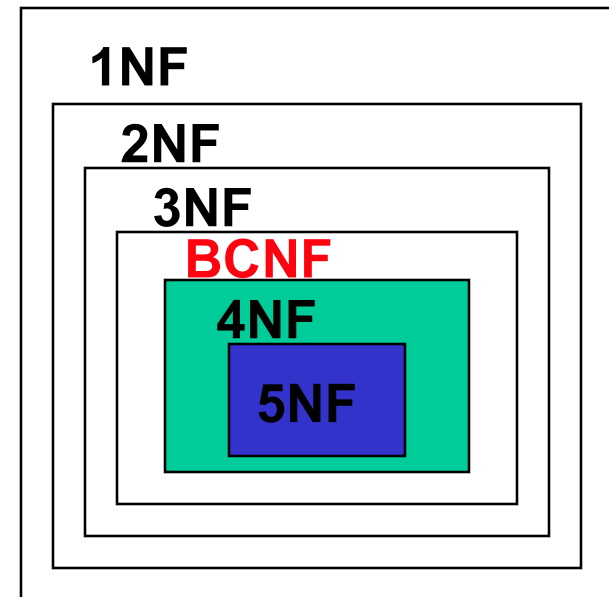
Relational Database Design

■ Introduction (con'd)

◆ *Normalization theory*

- based on **functional dependencies**

- * universe of relations
- * 1st Normal Form (1NF)
- * 2NF
- * 3NF
- * BCNF
- * 4NF
- * ...



Normalization

- Normalization
 - ◆ Proposed by Codd (1972)
 - ◆ take a relation schema through a series of tests based on **FD** and **primary keys** to certify whether it satisfies a certain **normal form**
 - ◆ Decompose a relation into several related relation to achieve:
 - Minimizing redundancy
 - Minimizing the insertion, deletion and update anomalies

Normalization

- Normalization requires two properties

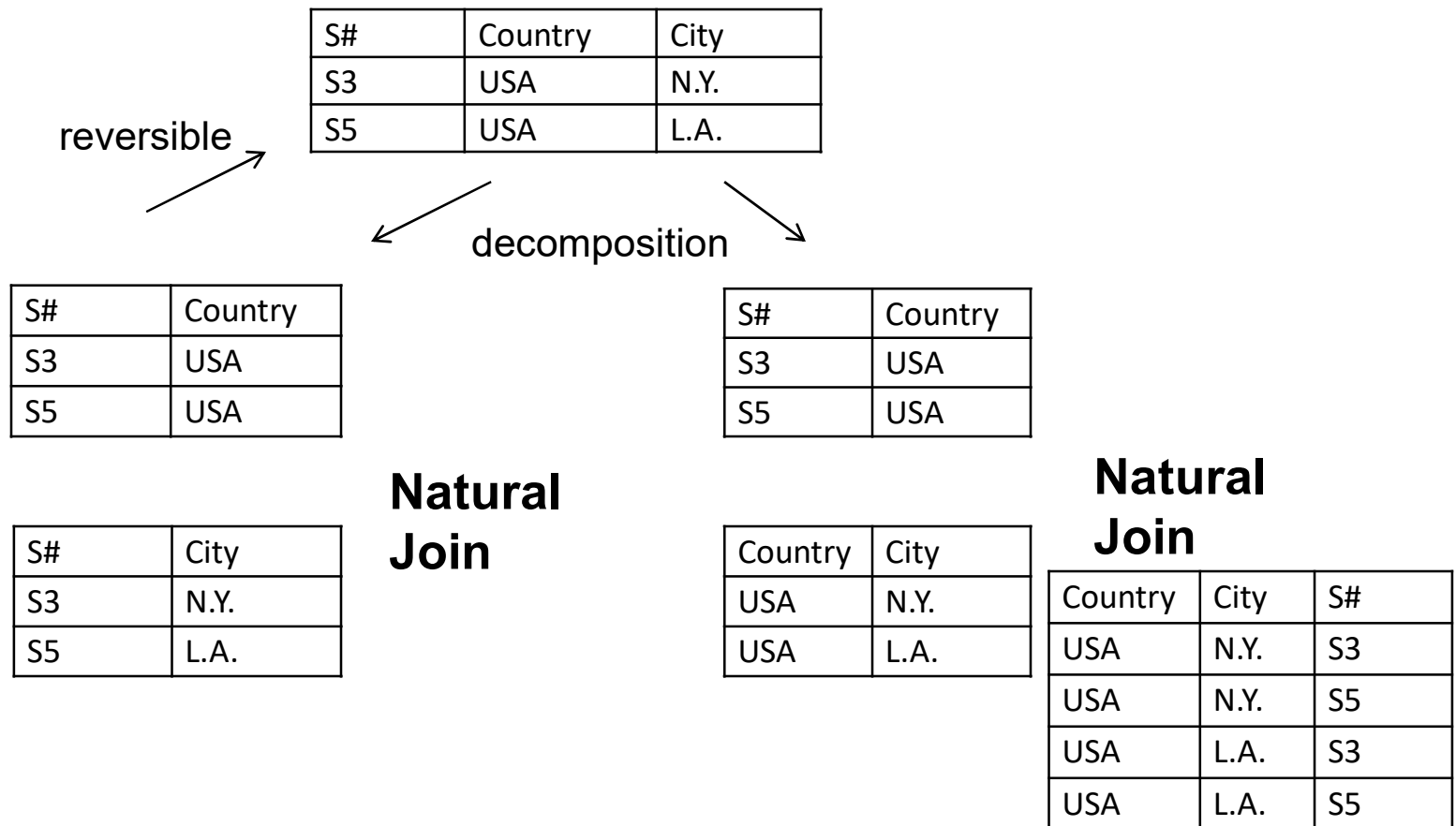
- ◆ Non-additive or lossless join

- Decomposition is **reversible** and no information is lost
 - No **spurious tuples** (tuples that should not exist) should be generated by doing a natural-join of any relations
(Extremely important)

- ◆ Preservation of the functional dependencies

- Ensure each functional dependency is represented in some individual relation or could be inferred from other dependencies after decomposition.

Lossless Decomposition




First Normal Form with Primary Key

- First normal form (1NF)
 - ◆ Disallow multi-values attributes, composite attributes and their combination
 - ◆ The domain of an attribute **must be atomic** (simple and indivisible) values
- DEPARTMENT: each department can have a number of locations
- 1NF: DEPT_LOCATIONS(Dnumber, Dlocation)
- 1NF also disallows multivalued attributes that are themselves composite

(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.

(a)

EMP_PROJ

Ssn	Ename	Projs	
		Pnumber	Hours

(b)

EMP_PROJ

Ssn	Ename	Pnumber	Hours
123456789	Smith, John B.	1	32.5
		2	7.5
666884444	Narayan, Ramesh K.	3	40.0
453453453	English, Joyce A.	1	20.0
		2	20.0
333445555	Wong, Franklin T.	2	10.0
		3	10.0
		10	10.0
		20	10.0
999887777	Zelaya, Alicia J.	30	30.0
		10	10.0
987987987	Jabbar, Ahmad V.	10	35.0
		30	5.0
987654321	Wallace, Jennifer S.	30	20.0
		20	15.0
888665555	Borg, James E.	20	NULL

(c)

EMP_PROJ1

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

EMP_PROJ2

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

Figure 14.10 Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a nested relation attribute PROJS. (b) Sample extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

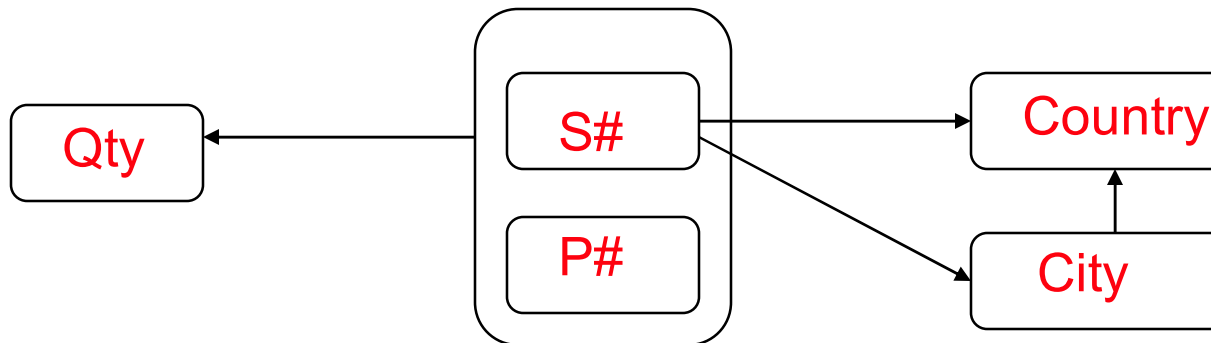
Problems with 1NF

◆ Example:

FIRST(S#, Country, City, P#, Qty)

and its Functional Dependency Diagram:

What's the primary key? (S#,P#)



Problems with 1NF

■ Problems with 1NF

◆ Insert Anomalies

◆ Inability to represent certain information:

Eg, cannot enter “Supplier and City” information until Supplier supplies at least one product

◆ Delete Anomalies

◆ Deleting the “only tuple” for a supplier will destroy all the information about that supplier

◆ Update Anomalies

◆ “S# and City” could be redundantly represented for each P#, which may cause potential inconsistency when updating a tuple

Solution: 1NF->2NF

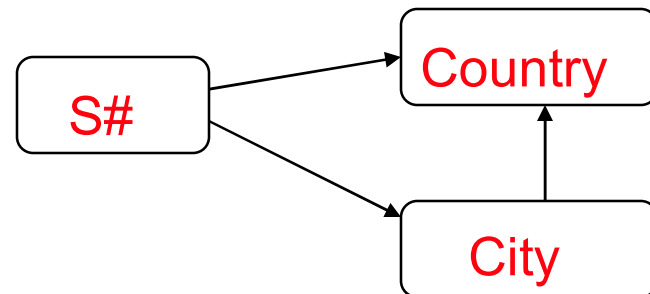
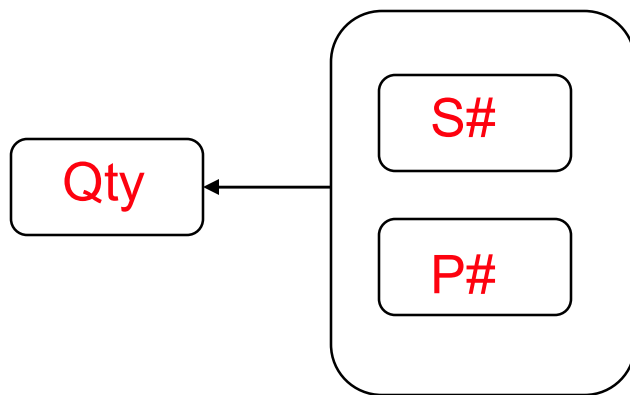
■ Possible Solution:

Replace the original table by two sub-tables

SECOND(S#, City, Country)

SP (S#, P#, Qty)

and now, their FD diagrams are:



Second Normal Form with Primary Key

- $X \rightarrow Y$
- Full functional dependency
 - ◆ If **removal** of any attribute A from X means that the dependency does not hold any more
 - ◆ E.g., $\{S\#, P\# \} \rightarrow Qty$
- Partial functional dependency
 - ◆ If some attributes A belonging to X **can be removed** from X and the dependency still holds
 - ◆ E.g., $\{S\#, P\# \} \rightarrow Country$ as $S\# \rightarrow Country$

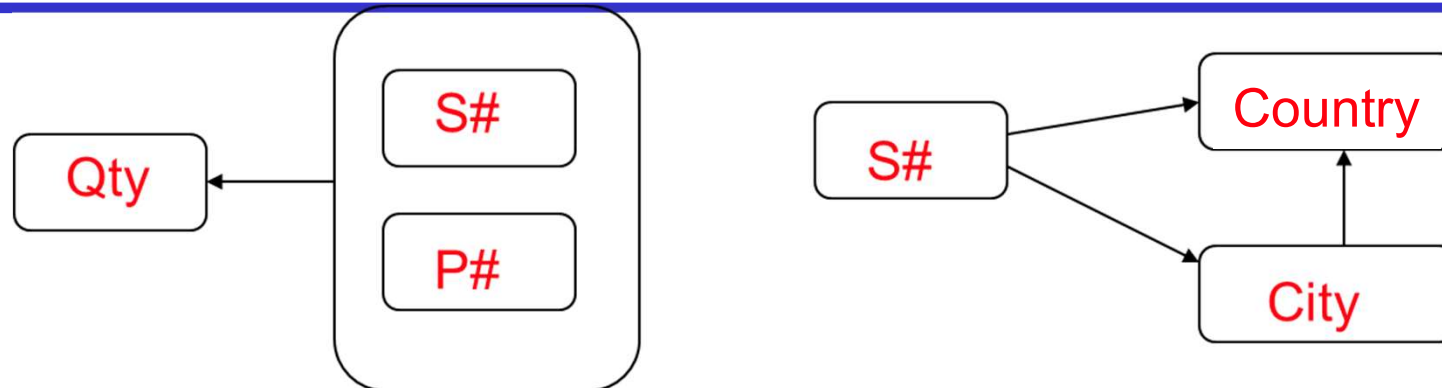
Second Normal Form with Primary Key

- A relation R is in 2NF if every **nonprime attributes** A in R is **fully functional dependent** on the **primary key** of R
- An attribute of R is called **prime (key) attribute** of R if it is a member of some **candidate key** of R. Otherwise it is nonprime
- Non-prime (non-key) attributes: Country, City, Qty
- Primary key: {S#, P#}

Second Normal Form with Primary Key

- The nonprime attributes **Country** violates 2NF because of $S\# \rightarrow \text{Country}$
- Similarly, $S\# \rightarrow \text{City}$
- If a relation is not in 2NF, it can be 2NF normalized into a number of 2NF relations in which **nonprime attributes** are associated only with the part of the primary key on which they are **fully functional dependent**.
- Solutions: **SP** (P#, S#, Qty); **Second**(S#, City, Country);

Problem with 2NF



- Still, there are problems with 2NF:
 - ◆ Insert Anomalies
 - ◆ Cannot insert “City and Country” information until some Supplier in that city exist
 - ◆ Delete Anomalies
 - ◆ Deleting an “only tuple” for a particular city will destroy all the information about that city
 - ◆ Update Anomalies
 - ◆ The supplier move to another city, inconsistency between city and country will happen

Solution: 2NF->3NF

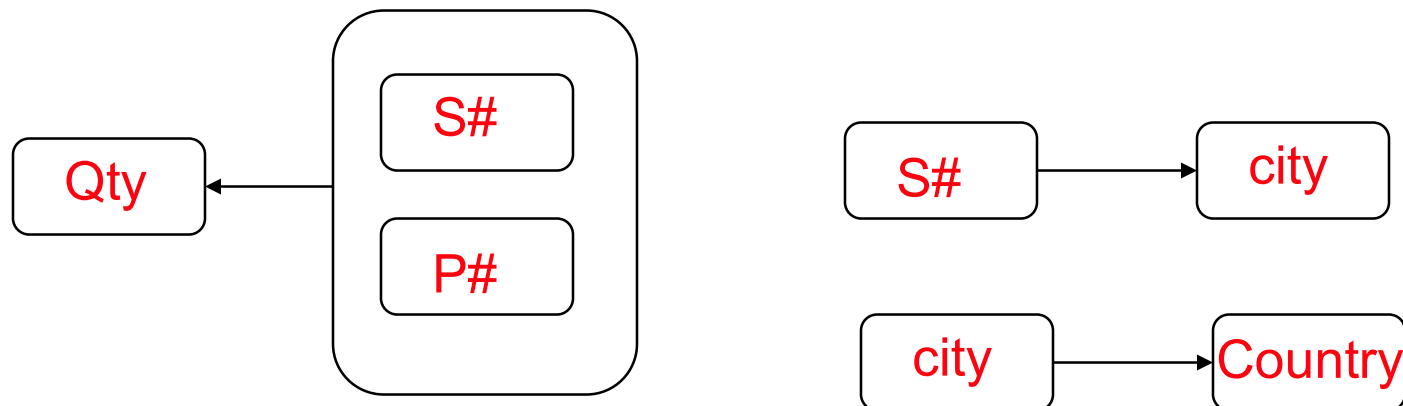
■ Possible Solution:

Replace the SECOND table by two sub-tables

SC (S#, City)

CC (City, country)

and still keep the table SP (S#, P#, Qty). Now the FD diagrams for the new tables:

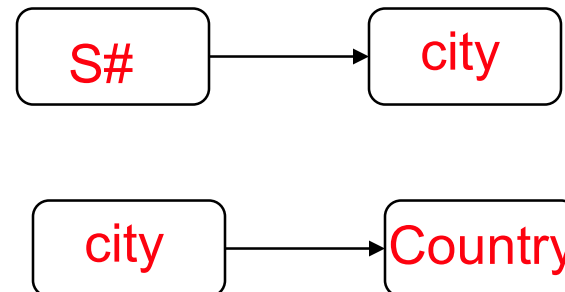
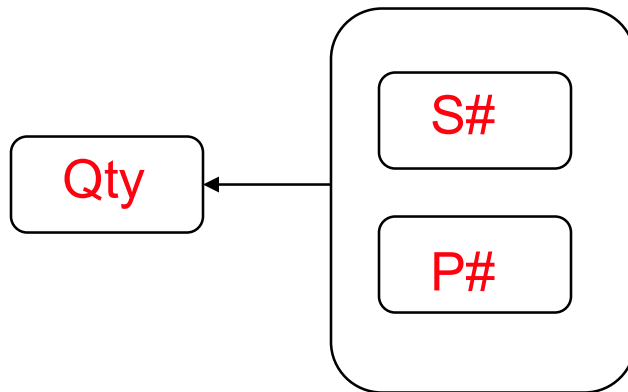


Third Normal Form with Primary Key

- 3NF is based on the concept of **transitive dependency**
- A functional dependency $X \rightarrow Y$ in a relation R is **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
- $X \rightarrow Z \rightarrow Y$ (Z : a non-prime attribute or a set including non-prime attributes)
- E.g., dependency $S\# \rightarrow \text{Country}$ is transitive through **City**
 - ◆ $S\# \rightarrow \text{City}$ and $\text{City} \rightarrow \text{Country}$, and **City** is neither a key itself nor a subset of the key
 - ◆ $S\# \rightarrow \text{City} \rightarrow \text{Country}$

Third Normal Form with Primary Key

- According to Codd's original definition, a relation scheme R is in 3NF if it satisfies 2NF and no **non-prime attribute of R** is **transitively dependent** on the **primary key**
- To normalize R into 3NF, R should be decomposed to break the transitive dependency.
- 3NF: SC (S#, City), CC (City, Country), SP (S#, P#, Qty).



Normal Forms Defined Informally

- 1st normal form
 - ◆ All attributes are simple (no composite or multivalued attributes)
- 2nd normal form
 - ◆ All non-prime attributes depend on the whole key (no partial dependency)
- 3rd normal form
 - ◆ All non-prime attributes only depend on the key (no transitive dependency)
- In general, 3NF is desirable and powerful enough
- But, still there are cases where a stronger normal form is needed

Test and Remedy for Normalization

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).

Consider all candidate keys of a relation (not just a defined primary key)

General Definitions of 2NF and 3NF

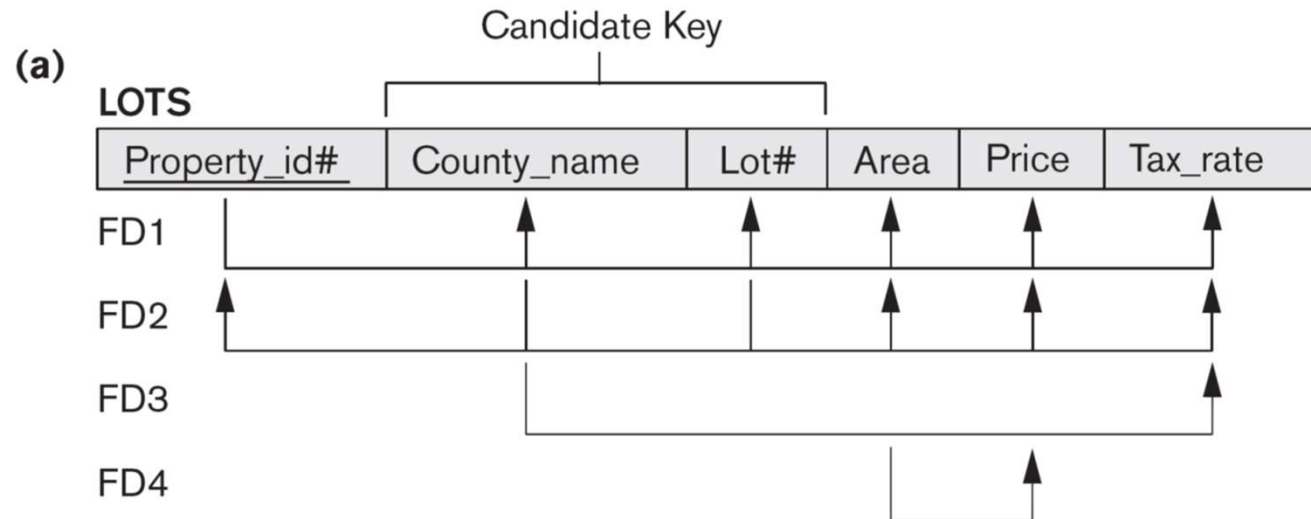
Definition. A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is not partially dependent on *any* key of R .¹¹

- **LOTS** describes parcels of land for sale in various counties of a state
- Two candidate keys: Property_id# and {County_name, Lot#}
- Property_id# is the primary key
- LOTS violates 2NF due to FD3 as Tax_rate is **partially dependent** on the **candidate key** {County_name, Lot#}
- County_name \Rightarrow Tax-rate (partial dependent)

General Definition of Second Normal Form

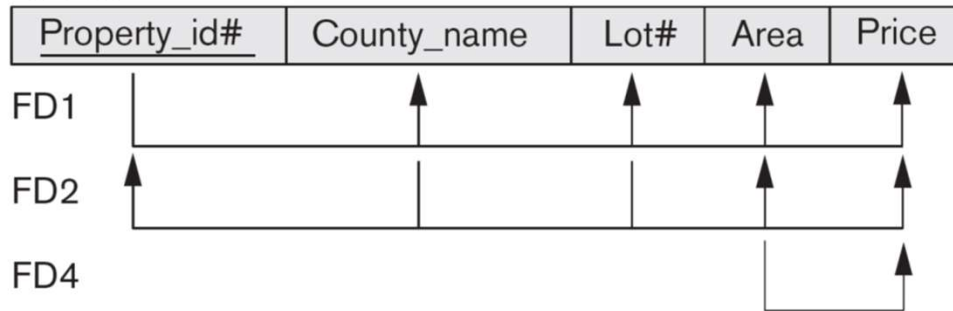
Figure 14.12

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) Summary of the progressive normalization of LOTS.

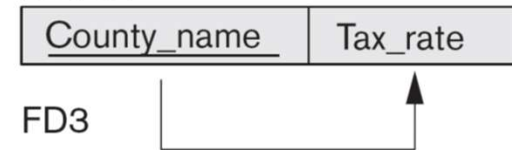


(b)

LOTS1

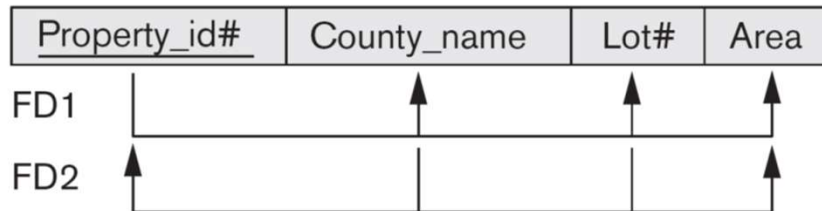


LOTS2

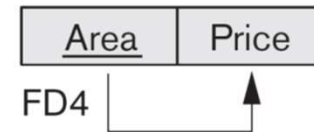


(c)

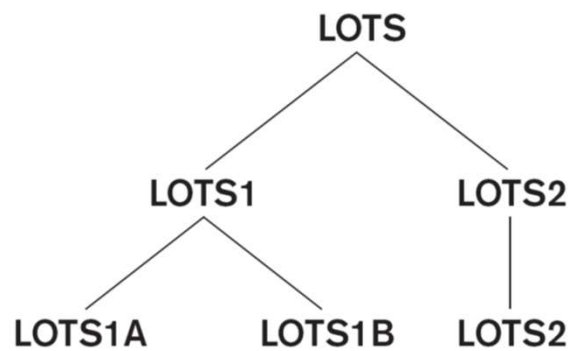
LOTS1A



LOTS1B



(d)



1NF

2NF

3NF

General Definition of Third Normal Form

Definition. A relation schema R is in **third normal form (3NF)** if, whenever a *nontrivial* functional dependency $X \rightarrow A$ holds in R , either (a) X is a superkey of R , or (b) A is a prime attribute of R .

- Superkey => include a key
- Prime attributes => attributes in a key
- LOTS2 is in 3NF
- FD4 in LOTS1 violates 3NF because Area is not a **superkey** and Price is **not a prime attribute** in LOTS1
- LOTS1 violates 3NF because Price is **transitively dependent** on each of the candidate keys of LOTS via the nonprime attribute Area
- To normalize LOTS1 into 3NF, LOTS1B{Area, Price}

Boyce-Codd Normal Form

- BCNF was proposed as a simpler form of 3NF, but it was found to be stricter than 3NF
- Every relation in BCNF is also in 3NF
 - ◆ BUT Relation in 3NF is not necessarily in BCNF

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 .

- Difference:
 - ◆ Condition which allows A to be prime is absent from BCNF

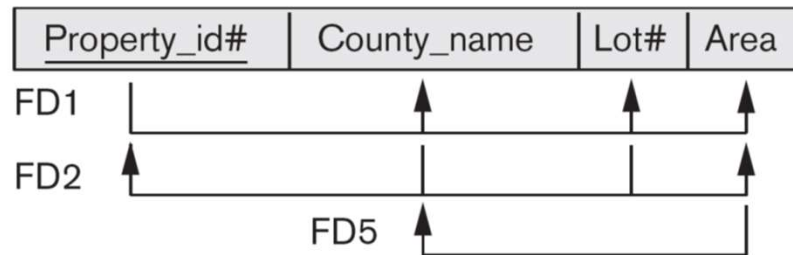
Definition. A relation schema R is in third normal form (3NF) if, whenever a *nontrivial* functional dependency $X \rightarrow A$ holds in R , either (a) X is a superkey of R , or (b) A is a prime attribute of R .

Boyce-Codd Normal Form

- Suppose we have thousands of lots in the relation but the lots are from only two counties: A and B
- Lot sizes in A are only 0.5, 0.6, 0.7, 0.8, 0.9 acres whereas lots in B are 1.1, 1.2, ..., 1.9
- Then, Area \rightarrow County_name
- FD5 satisfies 3NF in LOTS1A because County_name is a prime attribute
- FD5 violates BCNF in LOTS1A because Area is not a superkey of LOTS1A
- Create new relation R(Area, County_name). The area of lots can be represented in R(Area, County_name) to reduce redundancy

(a)

LOTS1A



BCNF Normalization

LOTS1AX

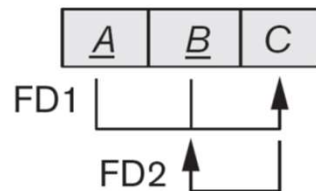


LOTS1AY



(b)

R



County_name is a prime attribute

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.

Boyce-Codd Normal Form

- Some notes on BCNF:
 - ◆ BCNF is stronger than 3NF
 - ◆ BCNF is useful when a relation has:
 - ◆ multiple candidate keys, and
 - ◆ these candidate keys are composite ones, and
 - ◆ they overlap on some common attributes
 - ◆ BCNF reduces to 3NF if the above conditions do not apply

References

- 6e
 - *Ch. 14, p. 487-515*
 - *Ch. 15, p. 529-533*