Basics of Functional Dependencies and Normalization

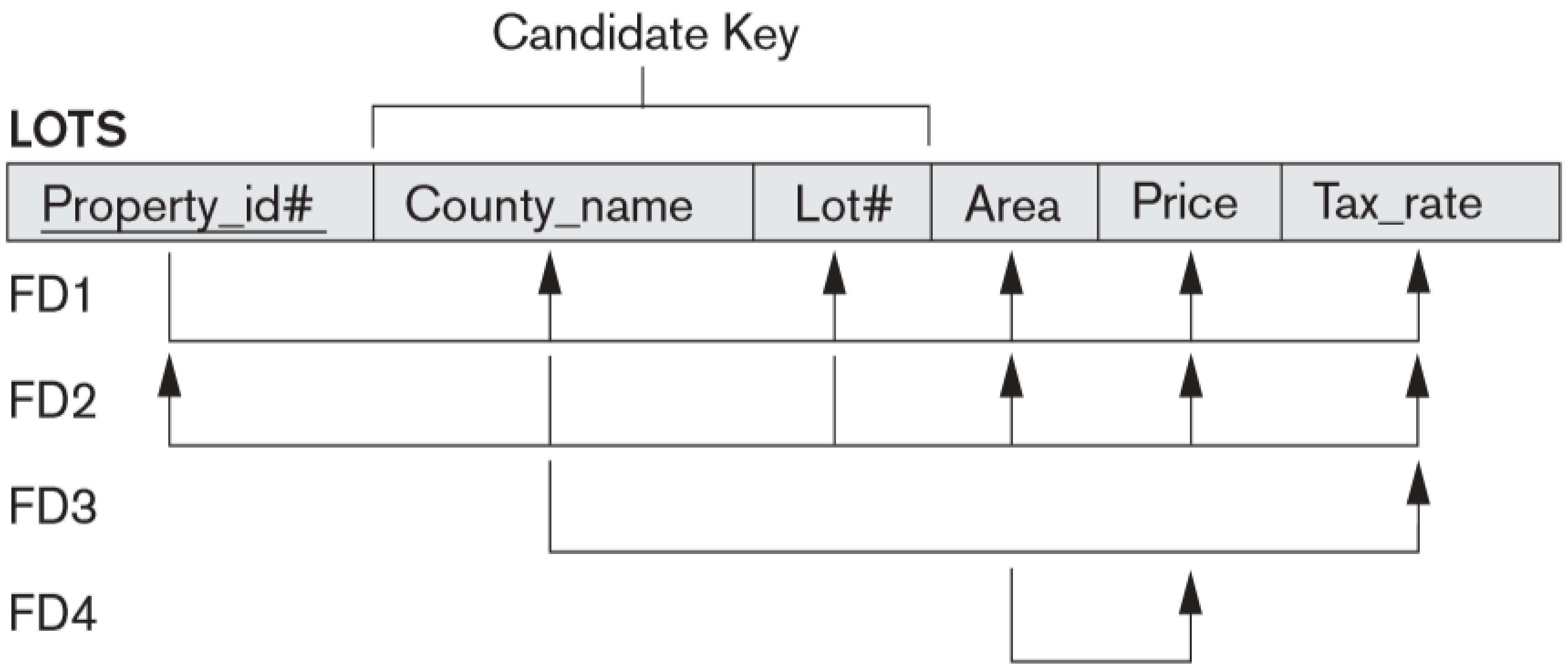
Part 4

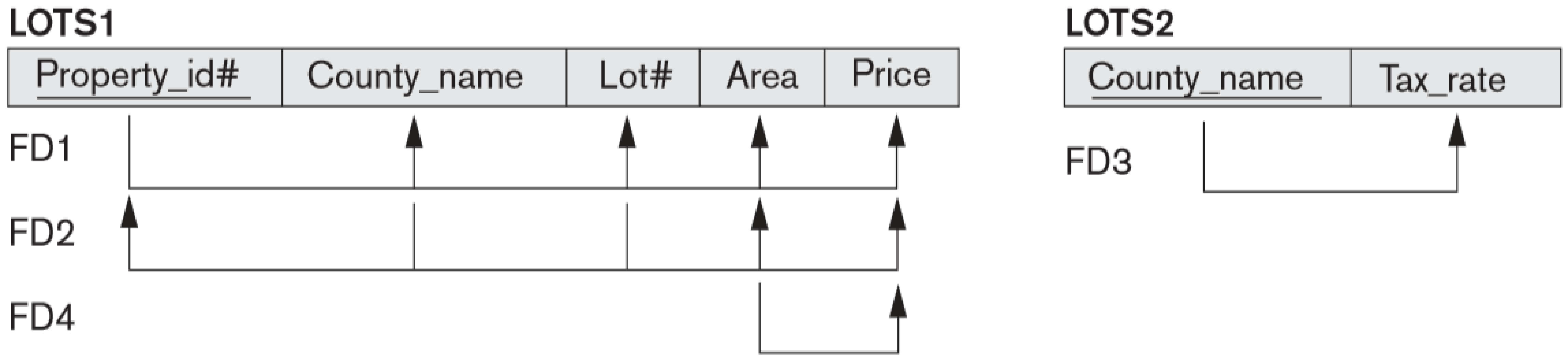
General Definitions of Second and Third Normal Forms

- As a general definition of **prime attribute**, an attribute that is part of *any candidate key* will be considered as prime.
- Partial and full functional dependencies and transitive dependencies will now be considered *with respect to all candidate keys* of a relation.

General Definition of Second Normal Form

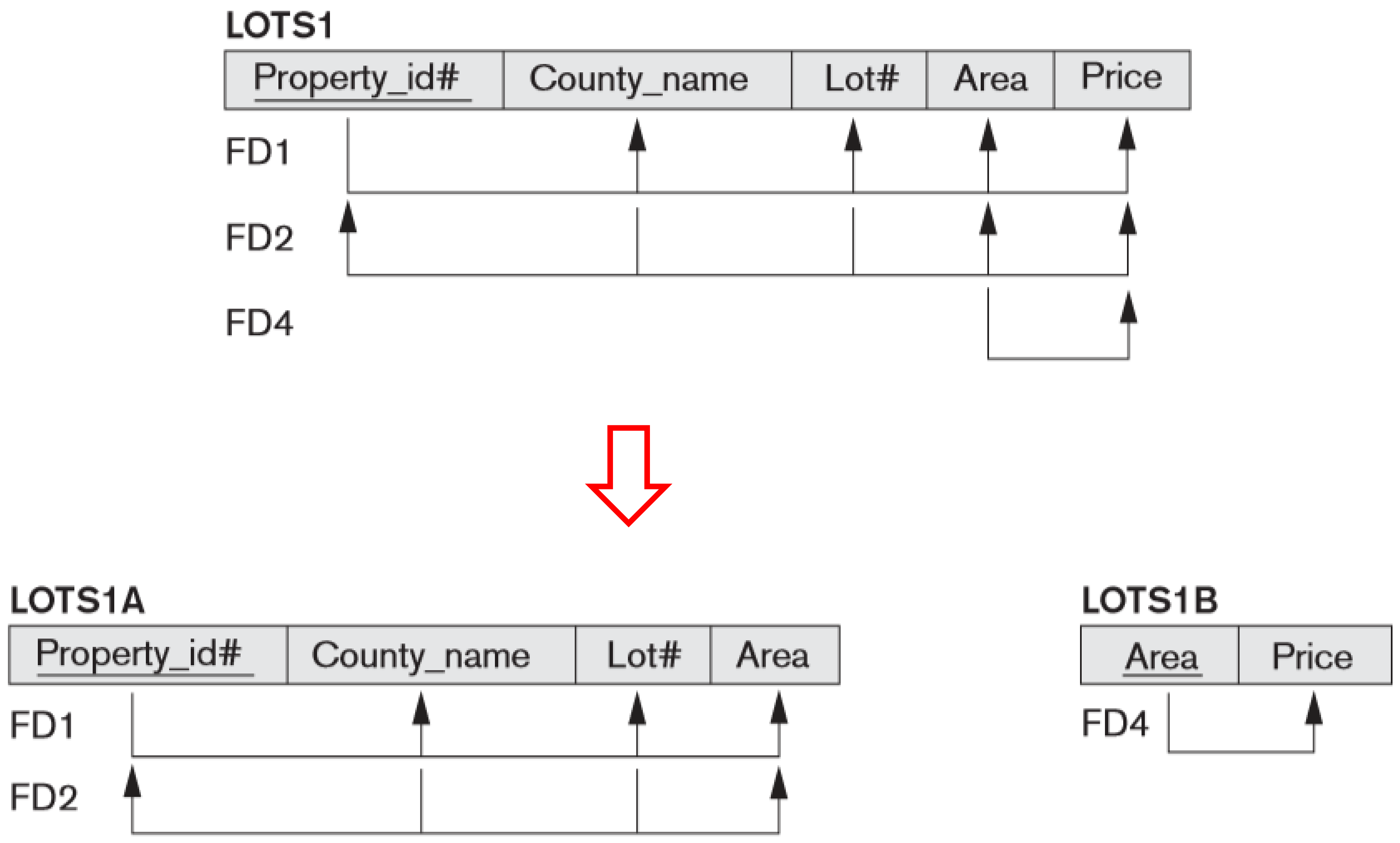
- **Definition.** A relation schema R is in **second normal form** (**2NF**) if every nonprime attribute A in R is not partially dependent on *any* key of R.


Candidate Key
LOTS
Property_id#
County_name
Lot#
Area
Price
Tax_rate
FD1
FD2
FD3
FD4


LOTS1
Property_id#
County_name
Lot#
Area
Price
FD1
FD2
FD4
LOTS2
County_name
Tax_rate
FD3

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.


LOTS1
Property_id#
County_name
Lot#
Area
Price
FD1
FD2
FD4
LOTS1A
Property_id#
County_name
Lot#
Area
FD1
FD2
LOTS1B
Area
Price
FD4

- Two points are worth noting about this example and the general definition of 3NF:
    - LOTS1 violates 3NF because Price is transitively dependent on each of the candidate keys of LOTS1 via the nonprime attribute Area.
    - This general definition can be applied *directly* to test whether a relation schema is in 3NF; it does *not* have to go through 2NF first. If we apply the above 3NF definition to `LOTS` with the dependencies FD1 through FD4, we find that *both* FD3 and FD4 violate 3NF. Therefore, we could decompose `LOTS` into `LOTS1A`, `LOTS1B`, and `LOTS2` directly. Hence, the transitive and partial dependencies that violate 3NF can be removed *in any order*.

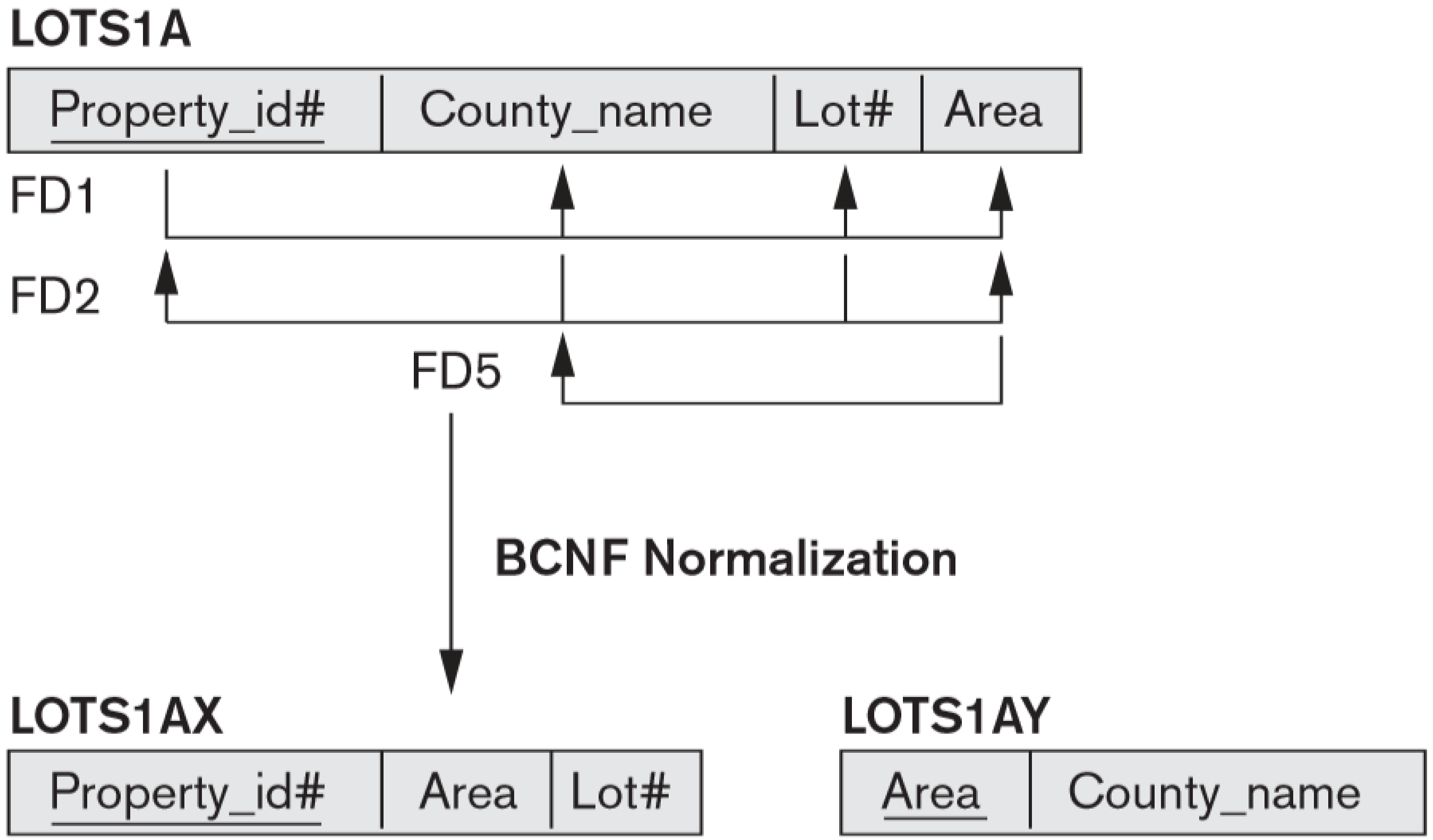
Interpreting the General Definition of Third Normal Form

- 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).

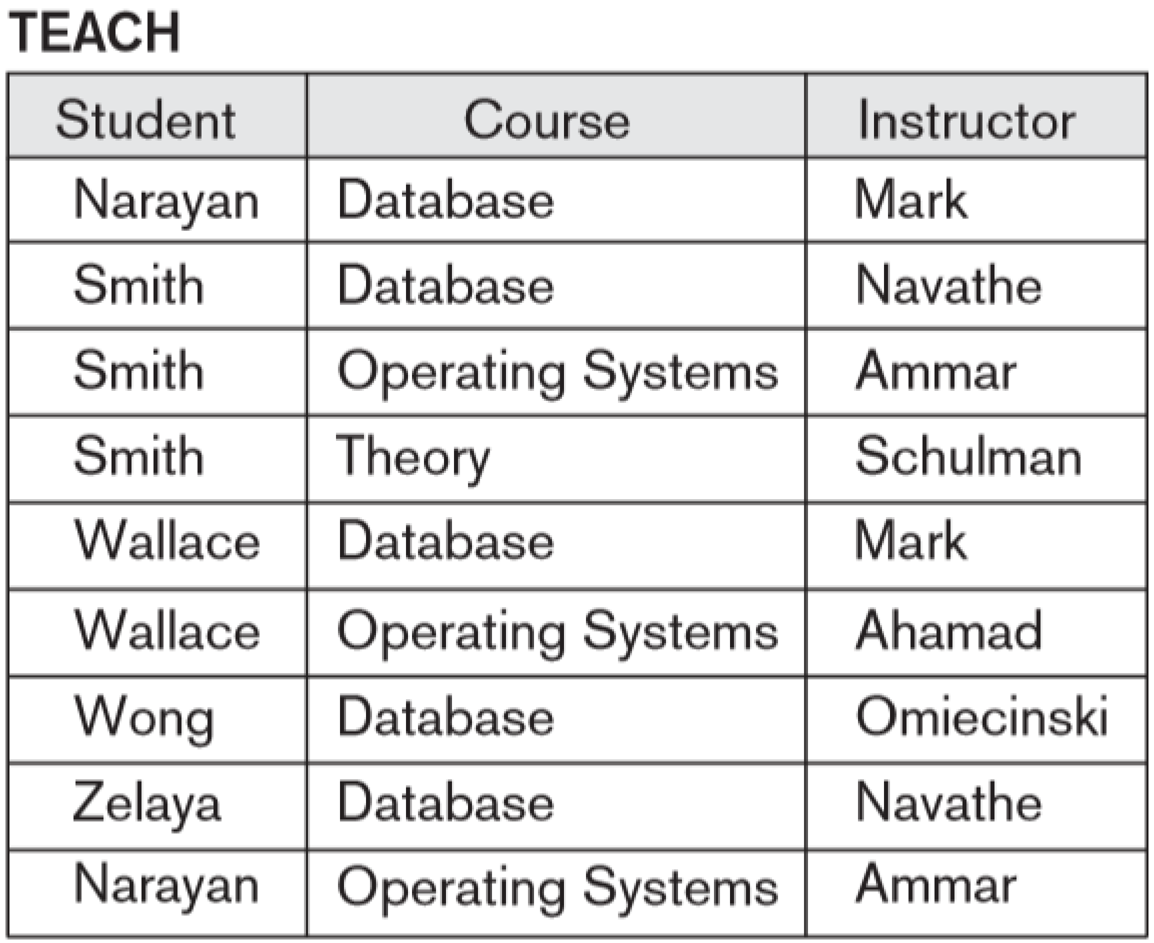
- Therefore, we can state a **general alternative definition of 3NF** as follows:
- **Alternative Definition.** A relation schema R is in 3NF if every nonprime attribute of R meets both of the following conditions:
  - It is fully functionally dependent on every key of R.
  - It is nontransitively dependent on every key of R.

Boyce-Codd Normal Form

- **Definition.** A relation schema R is in **BCNF** if whenever a *nontrivial* functional dependency $X \to A$ holds in R, then X is a superkey of R.


LOTS1A
Property_id#
County_name
Lot#
Area
FD1
FD2
FD5
BCNF Normalization
LOTS1AX
Property_id#
Area
Lot#
LOTS1AY
Area
County_name

- Another example



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

- FD1: $\{\texttt{Student}, \texttt{Course}\} \rightarrow \texttt{Instructor}$
  FD2: $\texttt{Instructor} \rightarrow \texttt{Course}$

- Note that {`Student`, `Course`} is a candidate key for this relation.
- 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.

Multivalued Dependency and Fourth Normal Form

- Multivalued dependencies are a consequence of first normal form (1NF),which disallows an attribute in a tuple to have a set of values, and the accompanying process of converting an unnormalized relation into 1NF.
- 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.

- 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.

EMP

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

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 $X \twoheadrightarrow Y \mid Z$.

- An MVD $X \twoheadrightarrow Y$ in R is called a **trivial MVD** if (a) Y is a subset of X, or (b) $X \cup Y = R$.
- 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.

- Notice that relations containing nontrivial MVDs tend to be **all-key relations**—that is, their key is all their attributes taken together.
- Furthermore, it is rare that such all-key relations with a combinatorial occurrence of repeated values would be designed in practice.
- However, recognition of MVDs as a potential problematic dependency is essential in relational design.

- **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.

EMP

Ename	Pname	Dname
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John

⇩

EMP_PROJECTS

Ename	Pname
Smith	X
Smith	Y

EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

Join Dependencies and Fifth Normal Form

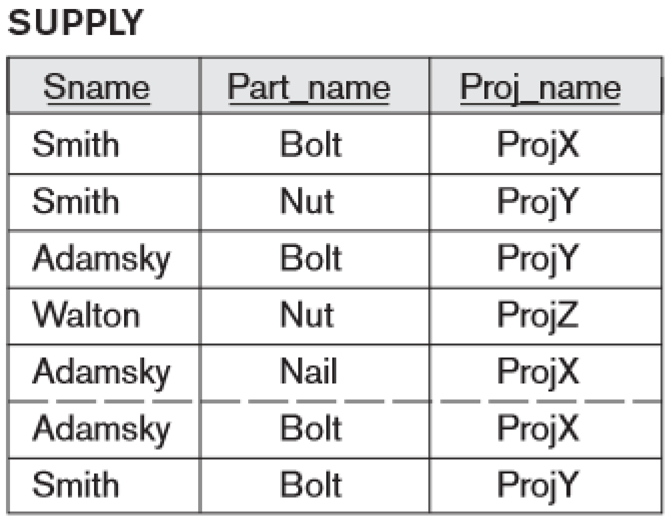
- **Definition.** A **join dependency** (**JD**), denoted by JD(R_1, R_2, ..., R_n), specified on relation schema R, specifies a constraint on the states r of R. The constraint states that every legal state r of R should have a nonadditive join decomposition into R_1, R_2, ..., R_n. Hence, for every such r we have

$$*(\pi_{R1}(r), \pi_{R2}(r), \dots, \pi_{Rn}(r)) = r$$

- Notice that an MVD is a special case of a JD where $n = 2$.
  - That is, a JD denoted as JD(R_1, R_2) implies an MVD ($R_1 \cap R_2$) $\twoheadrightarrow$ ($R_1 - R_2$) (or, by symmetry, ($R_1 \cap R_2$) $\twoheadrightarrow$ ($R_2 - R_1$)).
- A join dependency JD(R_1, R_2, ..., 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, ..., 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.

- **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, ..., R_n) in F^+ (that is, implied by F), every R_i is a superkey of R.

- Consider the `SUPPLY` all-key relation.
- 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.
- If this constraint holds, the tuples below the dashed line must exist in any legal state of the `SUPPLY` relation that also contains the tuples above the dashed line.



SUPPLY

Sname	Part_name	Proj_name
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

- The SUPPLY relation with the join dependency is decomposed into three relations R_1, R_2, and R_3 that are each in 5NF.

R_1

Sname	Part_name
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R_2

Sname	Proj_name
Smith	ProjX
Smith	ProjY
Adamsky	ProjY
Walton	ProjZ
Adamsky	ProjX

R_3

Part_name	Proj_name
Bolt	ProjX
Nut	ProjY
Bolt	ProjY
Nut	ProjZ
Nail	ProjX

- 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.

- Discovering JDs in practical databases with hundreds of attributes is next to impossible.
- It can be done only with a great degree of intuition about the data on the part of the designer.
- Therefore, the current practice of database design pays scant attention to them.