4CCS1DBS – Database Systems

Functional Dependencies and Normalisation for Relational Databases (2)

Re-cap: Informal Design Guidelines for Relational Databases

- What is relational database design?
  - The grouping of attributes to form "good" relation schemas
- Design is concerned mainly with base relations
- What are the criteria for "good" base relations?

Re-cap: Informal Design Guidelines - Semantics of Relation Attributes

- 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.
- So: *Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.*

Re-cap: Informal Design Guidelines – Redundant Information and Update Anomalies

- When information is stored redundantly:
  - Wastes storage
  - Causes problems with update anomalies
    - Insertion anomalies
    - Deletion anomalies
    - Modification anomalies
- GUIDELINE 2:
  - Design a schema that does not suffer from the insertion, deletion and update anomalies.
  - If any anomalies exist, note them so that applications can take them into account.

Re-cap: Informal Design Guidelines - Null Values in Tuples

- GUIDELINE 3:
  - 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)
- Example: if 10% of employees have individual offices, attribute should not be included in EMPLOYEE relation. Instead, use separate relation: EMP_OFFICES (ESSN, OFFICE-NUMBER).

Re-cap: Informal Design Guidelines - Spurious Tuples

- Bad designs for a relational database may result in *erroneous* results for certain JOIN operations

- GUIDELINE 4:
  - Design relation schemas that can be joined with equality conditions on attributes that are either primary keys or foreign keys in a way that guarantees that no spurious tuples will be generated.

  - Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations, as joining on such attributes will produce spurious tuples.

Recap: Functional Dependencies (1)

- Functional dependencies (FDs)
  - Used to specify *formal measures* of the "goodness" of relational designs
  - Keys are used to define **normal forms** for relations
  - **Constraints** are derived from the *meaning* and *interrelationships* of the data attributes
- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y

Recap: Functional Dependencies (2)

- Functional dependency (FD) between attributes X and Y of relations R:
  - **For any two tuples t1 and t2 in any relation instance r(R): If t1[X]=t2[X], *then* t1[Y]=t2[Y]**
  - **X → Y, if whenever two tuples have the same value for X, they *must have* the same value for Y**
- X → Y in R specifies a *constraint* on all relation instances r(R)
- Written as X → 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
- X: left-hand side, Y: right-hand side

Normal Forms Based on Primary Keys

- Normalisation of Relations
- Practical Use of Normal Forms
- Definitions of Keys and Attributes Participating in Keys
- First Normal Form
- Second Normal Form
- Third Normal Form

Normalisation of Relations (1)

- **Normalisation definition**
  - The process of:
    - decomposing unsatisfactory relations by breaking up their attributes into smaller relations
    - Analysing relation schemas based on FDs and primary keys to minimise redundancy and insertion, deletion and update anomalies
- **Normal form:**
  - Condition using ***keys*** and ***FDs*** of a relation to certify whether a relation schema is in a particular normal form

Normalisation 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 – not covered here
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation)

Practical Use of Normal Forms

- **Normalisation** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties

- The practical utility of these normal forms becomes questionable when the constraints on which they are based are *hard to understand* or to *detect*

- The database designers *need not* normalize to the highest possible normal form
  - (usually up to 3NF, BCNF or 4NF)

- **Denormalisation**:
  - The process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

Definitions of Keys and Attributes Participating in Keys (1)

- A ***superkey*** of a relation schema R = {A1, A2, ..., An} is a set of attributes S *subset-of*-R with the property that no two tuples t1 and t2 in any legal relation state r of R will have $t_1[S] = t_2[S]$

- 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, i.e. a key is a *<u>minimal</u>* superkey.

Key Constraints

- **Superkey** of R:
  - Is a set of attributes SK of R with the following condition:
    - No two tuples in any valid relation state r(R) will have the same value for SK
    - That is, for any distinct tuples t1 and t2 in r(R), t1[SK] $\neq$ t2[SK]
    - This condition must hold in *any valid state* r(R)
- **Key** of R:
  - A "minimal" superkey
  - That is, a key is a superkey K such that removal of any attribute from K results in a set of attributes that is not a superkey (does not possess the superkey uniqueness property)

Key Constraints (continued)

- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - Define superkeys of relation CAR
  - Define keys of relation CAR

Key Constraints (continued)

- Example: Consider the CAR relation schema:
  - CAR(State, Reg#, SerialNo, Make, Model, Year)
  - CAR has two keys:
    - Key1 = {State, Reg#}
    - Key2 = {SerialNo}
  - Both are also superkeys of CAR
  - {SerialNo, Make} is a superkey but *not* a key.
- In general:
  - Any *key* is a *superkey* (but not vice versa)
  - Any set of attributes that *includes a key* is a *superkey*
  - A *minimal* superkey is also a key

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**.

- ***Prime*** is an attribute that is member of *some* candidate key

- ***Nonprime*** is an attribute that is not a prime attribute - i.e. it is not a member of *any* candidate key.

First Normal Form

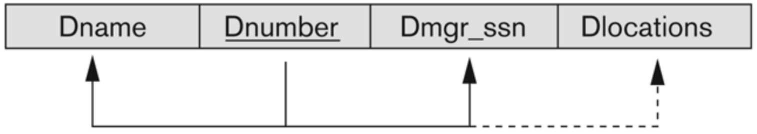
- Disallows
  - composite attributes
  - multivalued attributes
  - **nested relations**; attributes whose values for an *individual tuple* are non-atomic

- Considered to be part of the definition of relation.

Normalisation into 1NF

(a)

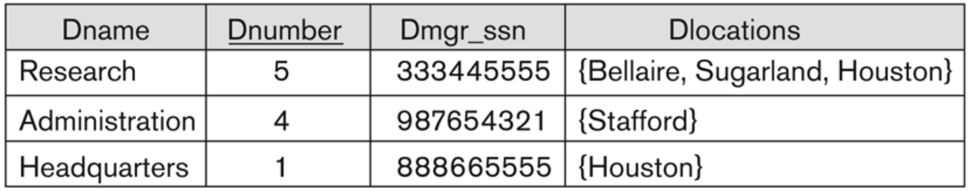
DEPARTMENT



Dname	Dnumber	Dmgr_ssn	Dlocations

(b)

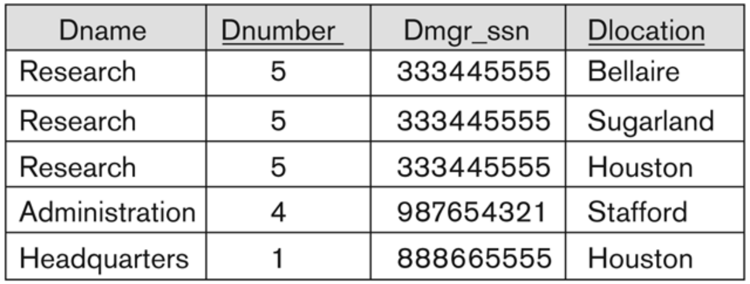
DEPARTMENT



Dname	Dnumber	Dmgr_ssn	Dlocations
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

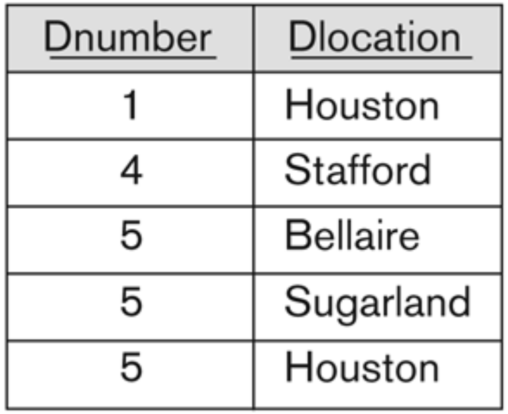
DEPARTMENT



Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

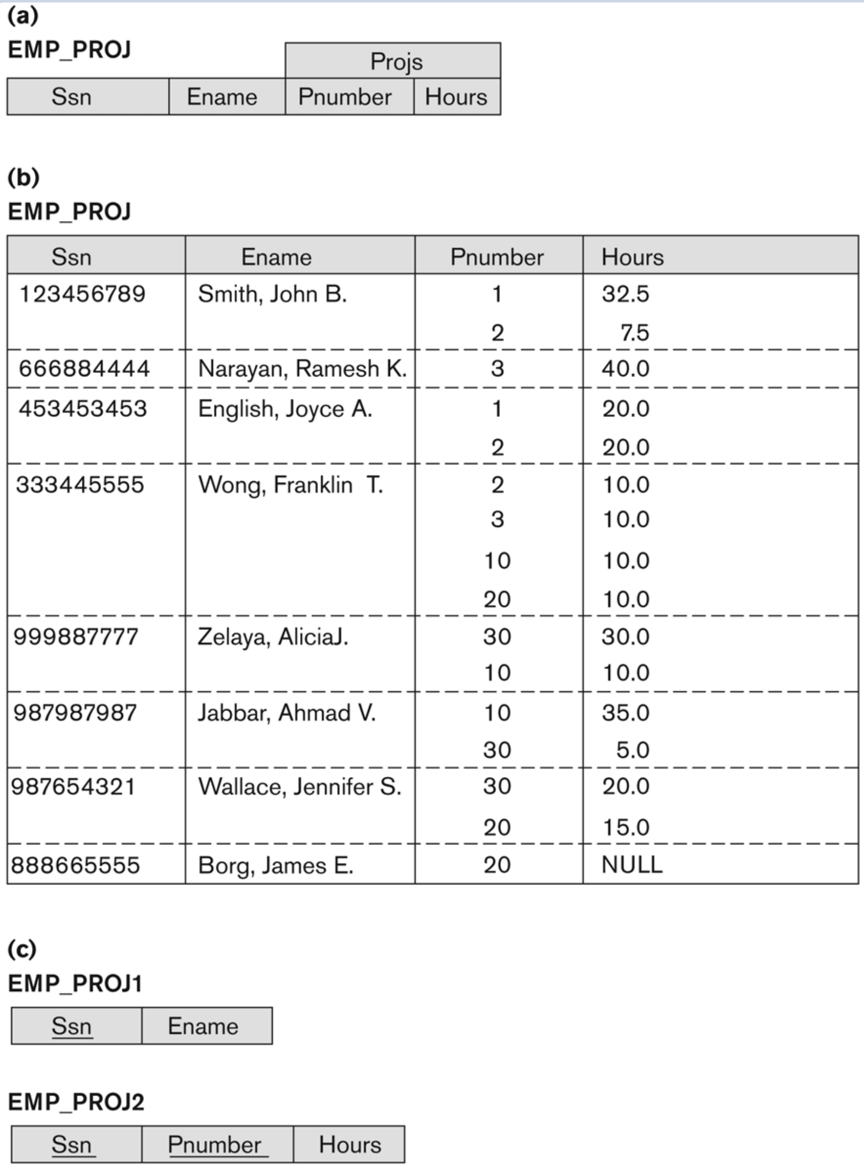
To normalise: remove attribute that causes the problem and place in separate relation together with the primary key

DEPT_LOCATIONS



Dnumber	Dlocation
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

Normalisation of Nested Relations into 1NF



(a)

EMP_PROJ

		Projs	
Ssn	Ename	Pnumber	Hours

(b)

EMP_PROJ

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

(c)

EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

Nested relations, as with composite attributes, are disallowed under 1NF.

Relation EMP_PROJ is NOT in 1NF.

Primary keys: SSN and Pnumber within nested relation.

To normalise: remove nested relation and place in separate relation together with the primary key, as in (c).

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→ Z where removal of any attribute from Y means the FD does not hold any more

Second Normal Form (2)

- Examples:
  - {SSN, PNUMBER} → HOURS is a full FD
    - since neither SSN → HOURS nor PNUMBER → HOURS hold
  - {SSN, PNUMBER} → ENAME is not a full FD
    - since SSN → ENAME also holds
    - it is called a *partial* dependency

Second Normal Form (3)

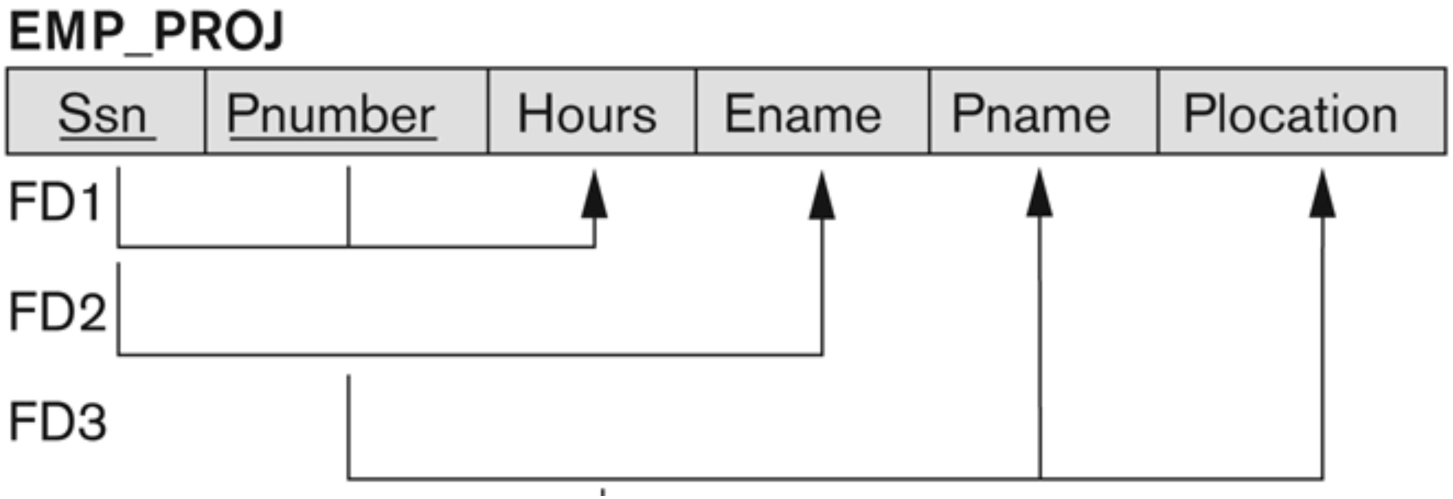
- A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is fully functionally dependent on the primary key

- Test that left-hand side in FD is part of 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 normalisation

Is this relation in 2NF?



EMP_PROJ

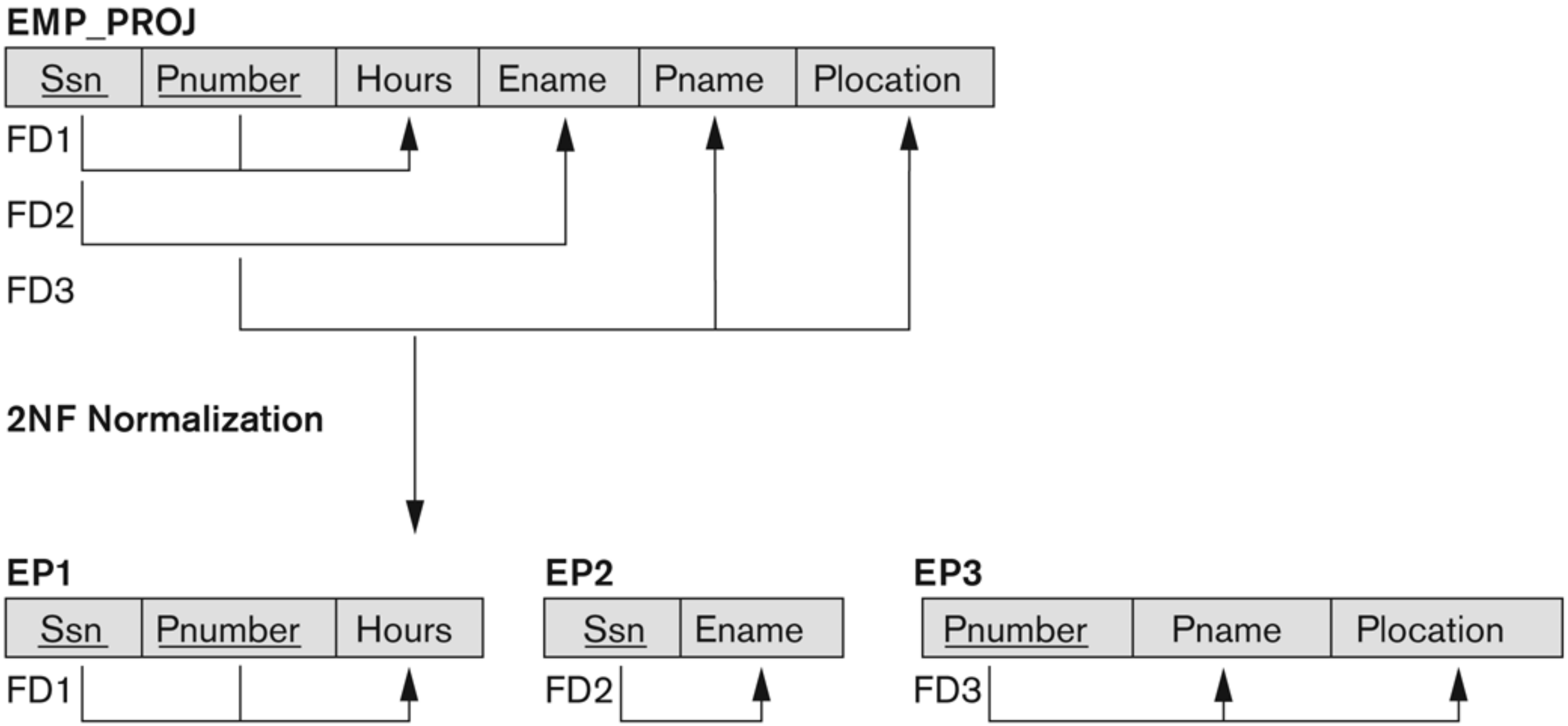
Ssn | Pnumber | Hours | Ename | Pname | Plocation

FD1

FD2

FD3

Normalising into 2NF



EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

FD1

FD2

FD3

2NF Normalization

EP1

Ssn	Pnumber	Hours

FD1

EP2

Ssn	Ename

FD2

EP3

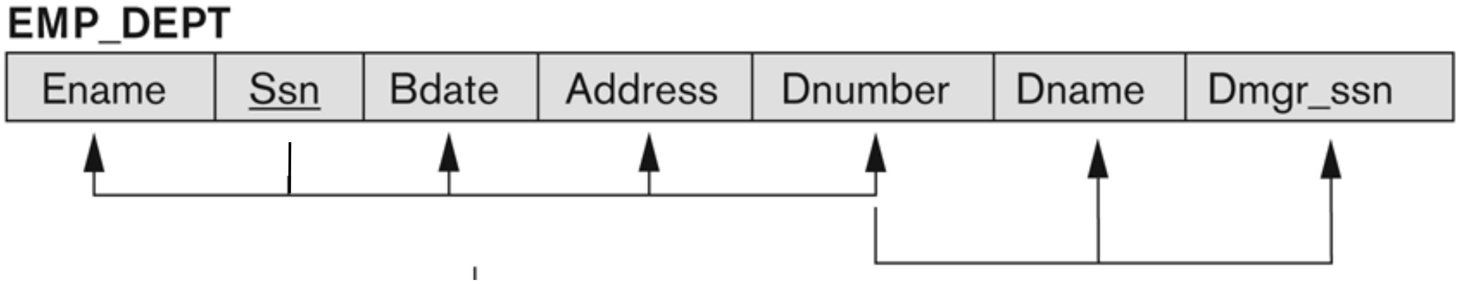
Pnumber	Pname	Plocation

FD3

Third Normal Form (1)

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

- **Examples:**
  - SSN → DMGRSSN is a **transitive** FD
    - SSN → DNUMBER and DNUMBER → DMGRSSN hold
  - SSN → ENAME is **non-transitive**
    - Since there is no set of attributes X where SSN → X and X→ ENAME



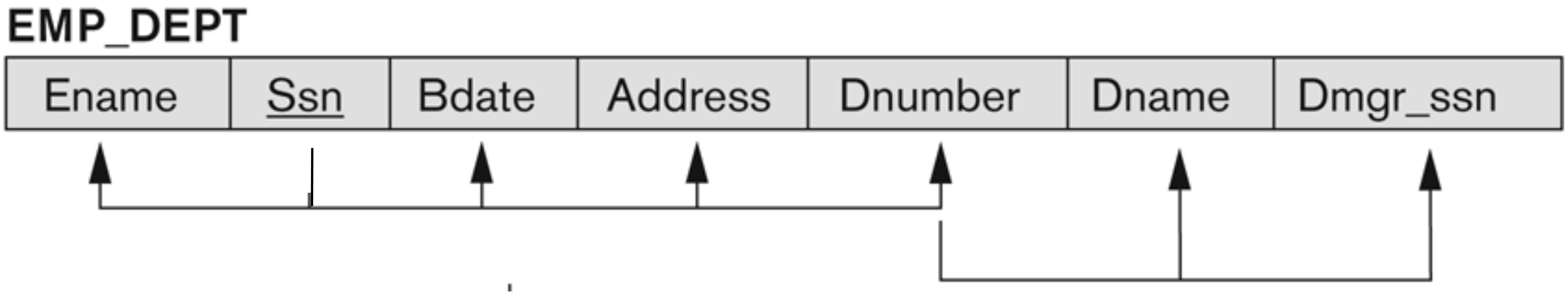
EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

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 normalisation
- NOTE:
  - In X → Y and Y → 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.

Is this relation in 3NF ?

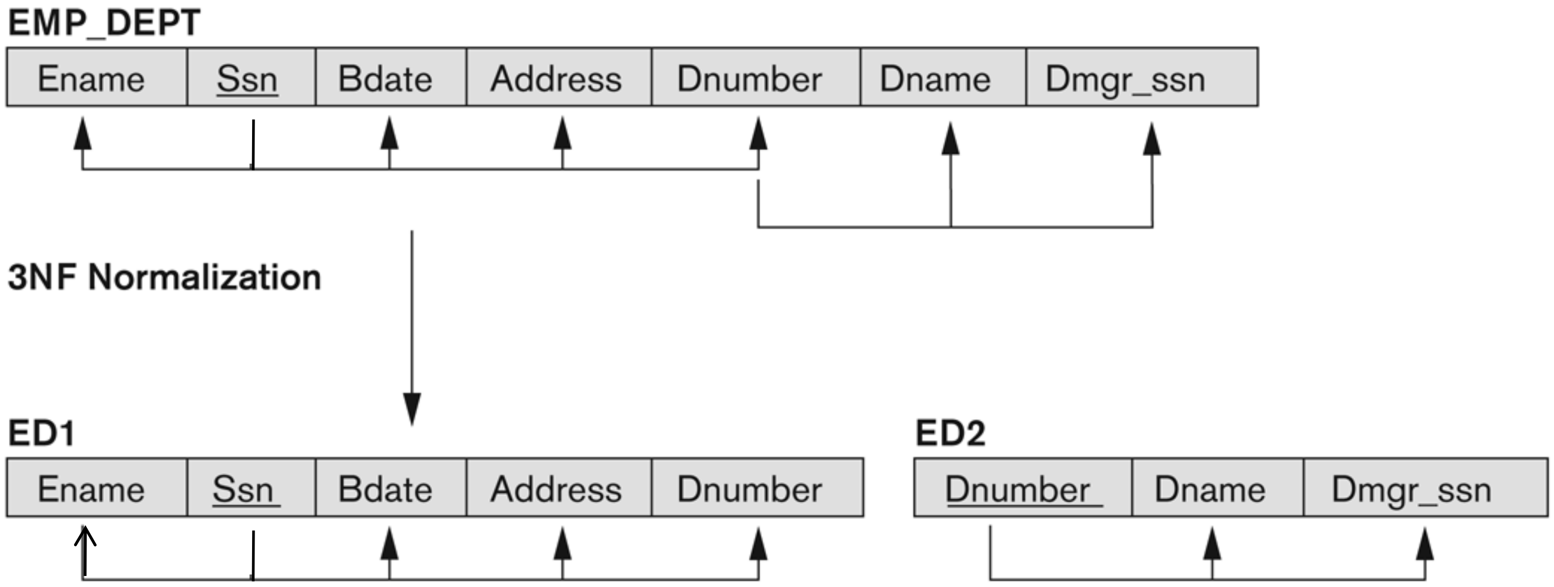


EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

- 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

Normalisation into 3NF



EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

3NF Normalization

ED1

Ename	Ssn	Bdate	Address	Dnumber

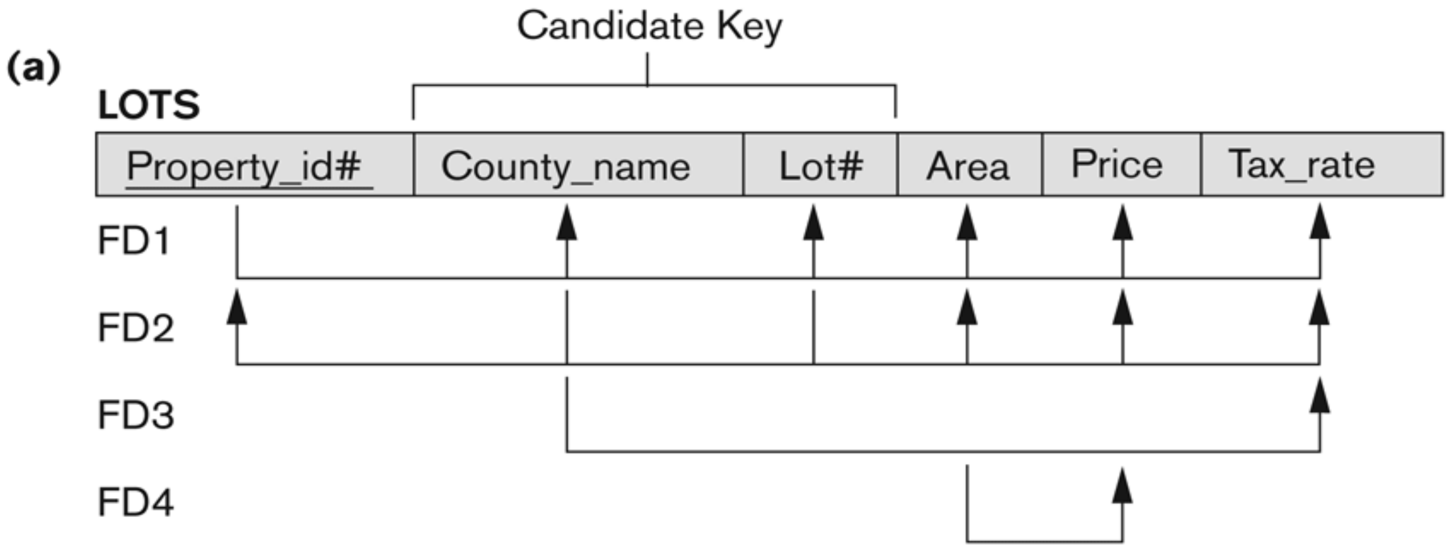
ED2

Dnumber	Dname	Dmgr_ssn

Normal Form Mnemonic ☺

- 1st normal form
  - All attributes depend on **the key**
- 2nd normal form
  - All attributes depend on **the whole key**
- 3rd normal form
  - All attributes depend on **nothing but the key**

The LOTS Relation

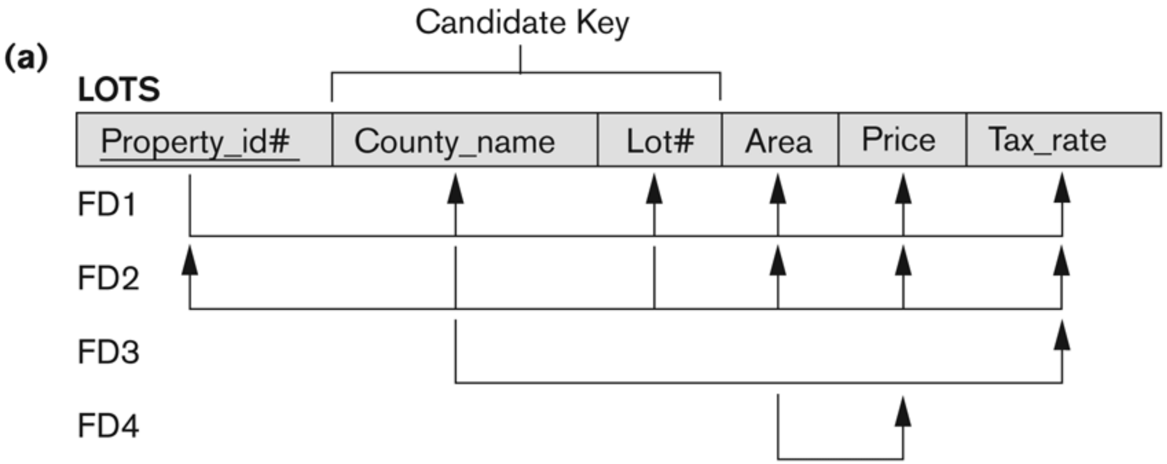


- FD3: tax rate is fixed for given county
- FD4: the price of a lot is determined by its area (regardless of which county it is in).

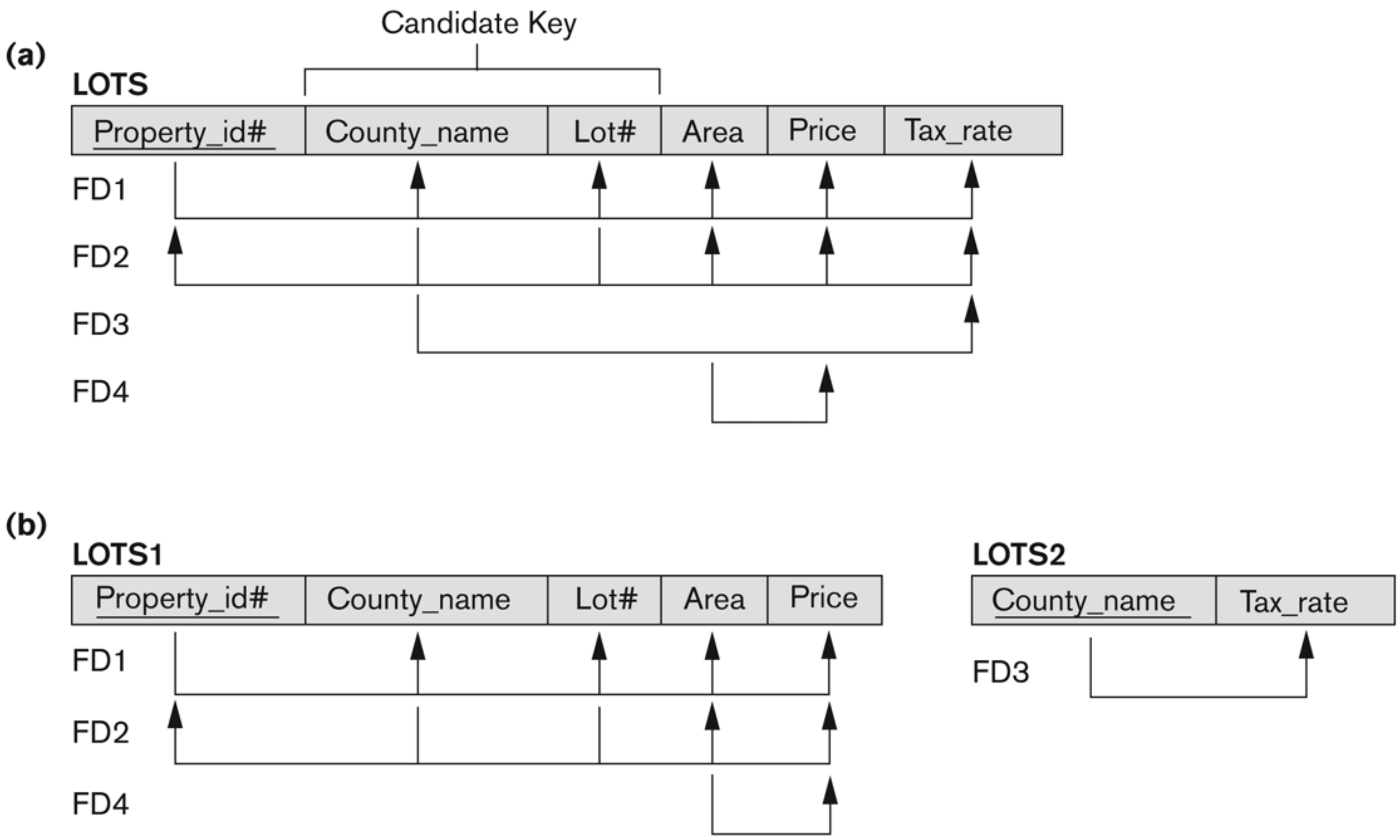
General Definitions for 2NF and 3NF

- 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.
  - Test for FDs whose left-had side attributes are ***part*** of the primary key.

Normalisation of LOTS into 2NF



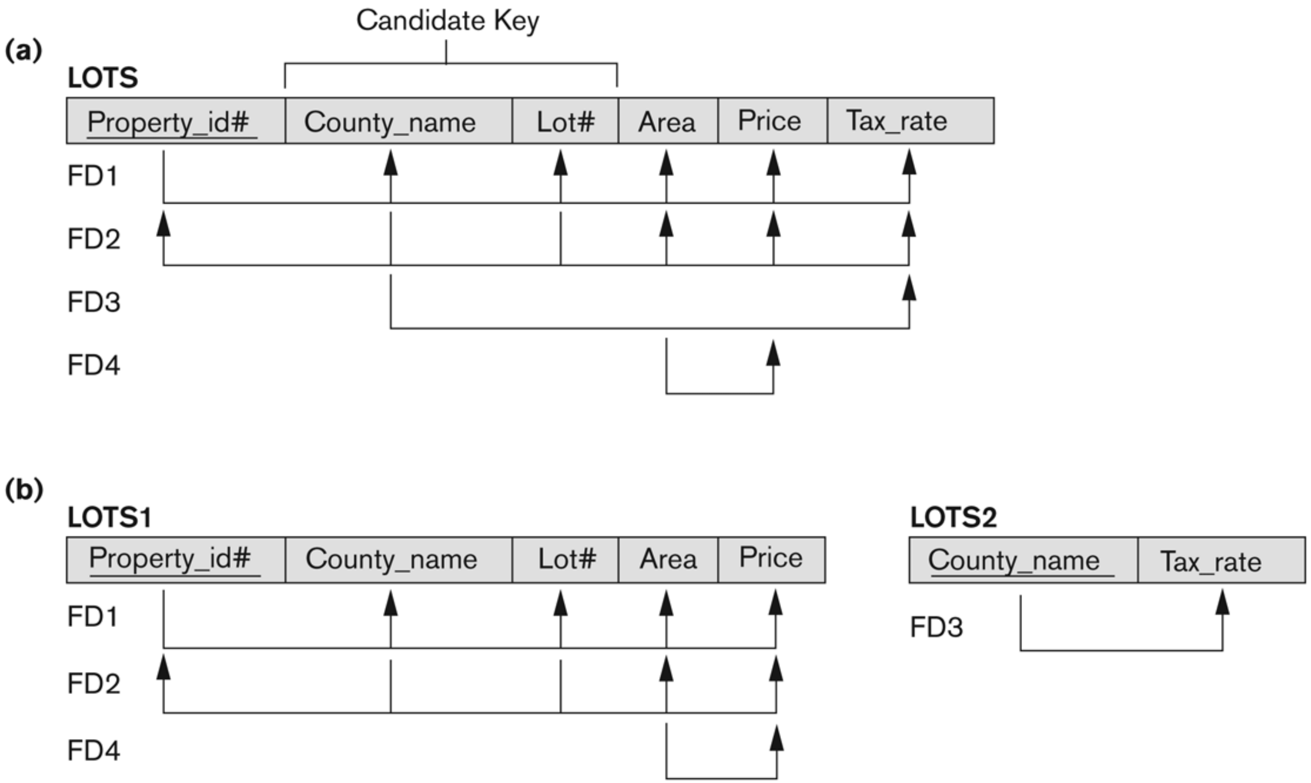
Normalisation of LOTS into 2NF



General Definitions for 2NF and 3NF

- A relation schema R is in **third normal form (3NF)** if, whenever a nontrivial FD X → A holds in R, either
- (a) X is a superkey of R, or
- (b) A is a prime attribute of R.

Normalisation of LOTS into 3NF



(a)

Candidate Key

LOTS

Property_id#	County_name	Lot#	Area	Price	Tax_rate

FD1

FD2

FD3

FD4

(b)

LOTS1

Property_id#	County_name	Lot#	Area	Price

FD1

FD2

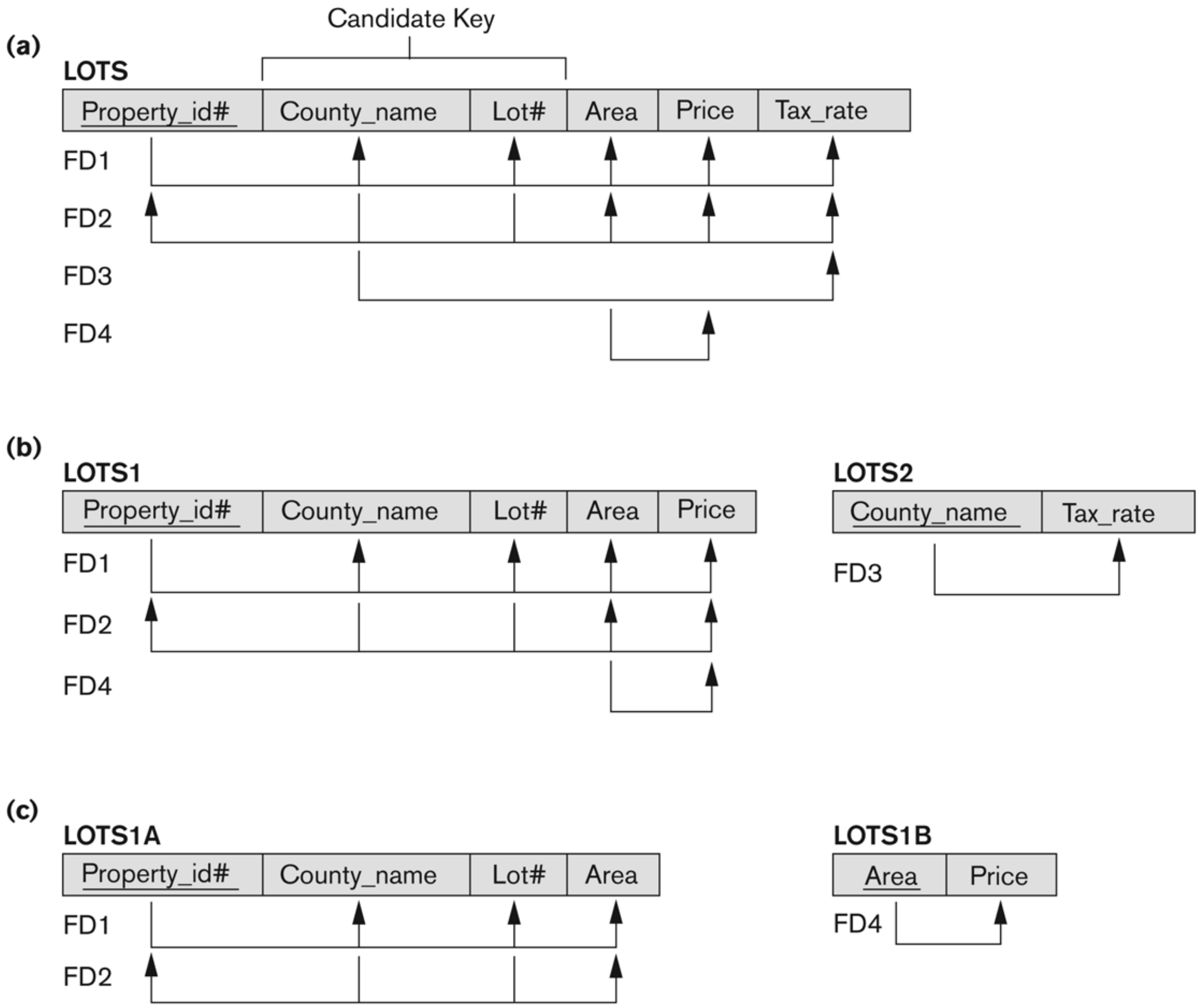
FD4

LOTS2

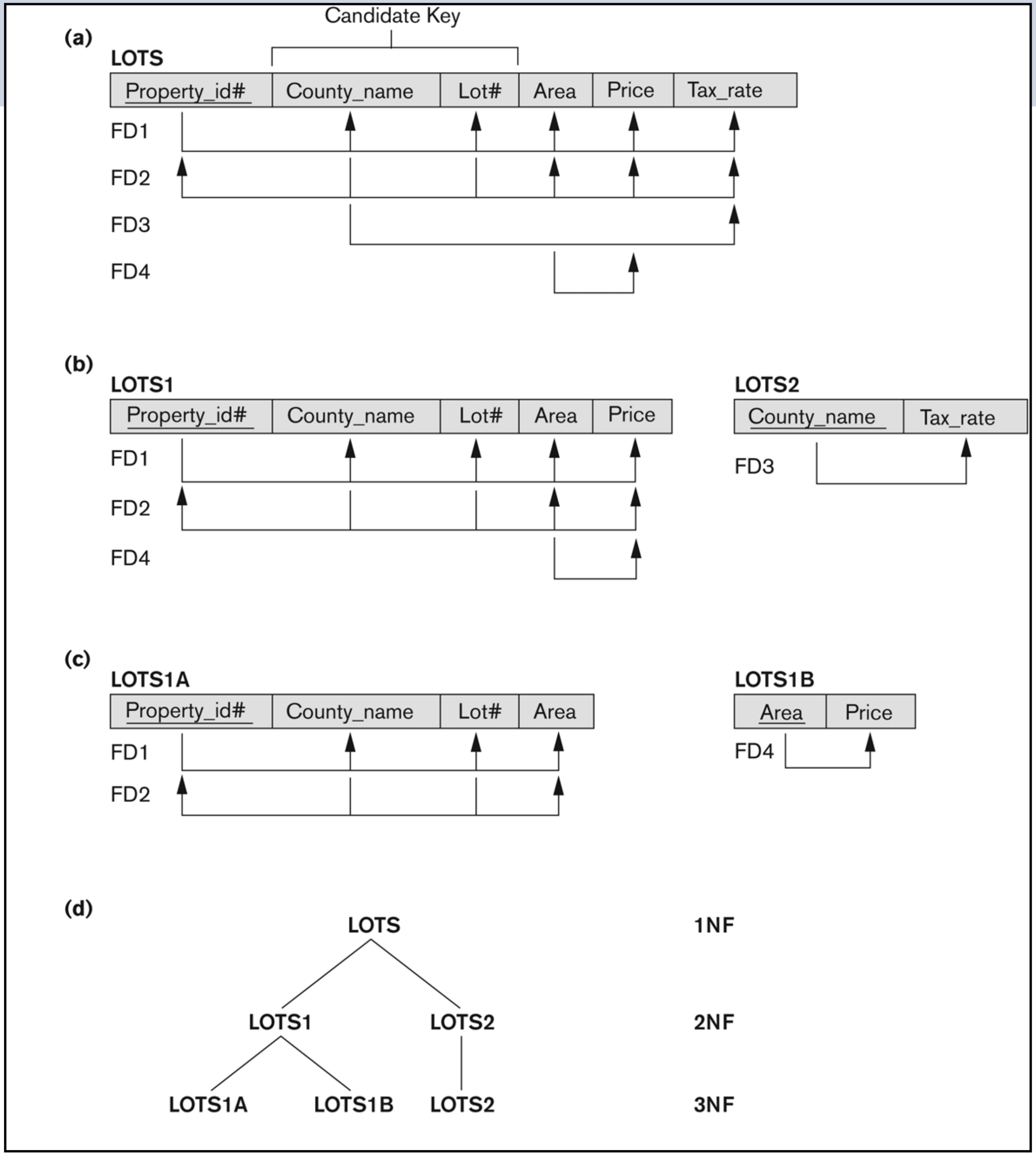
County_name	Tax_rate

FD3

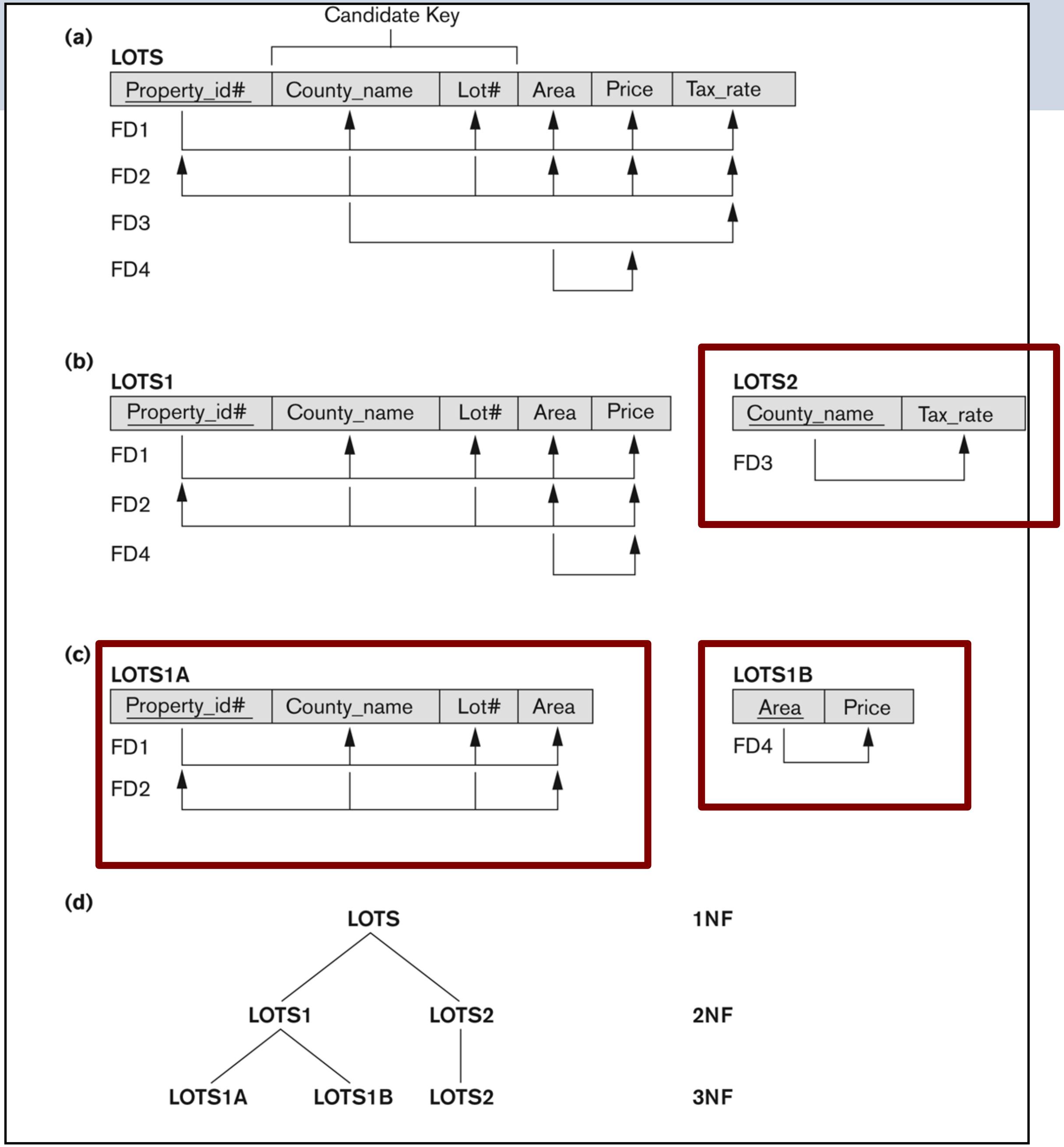
Normalisation of LOTS into 3NF



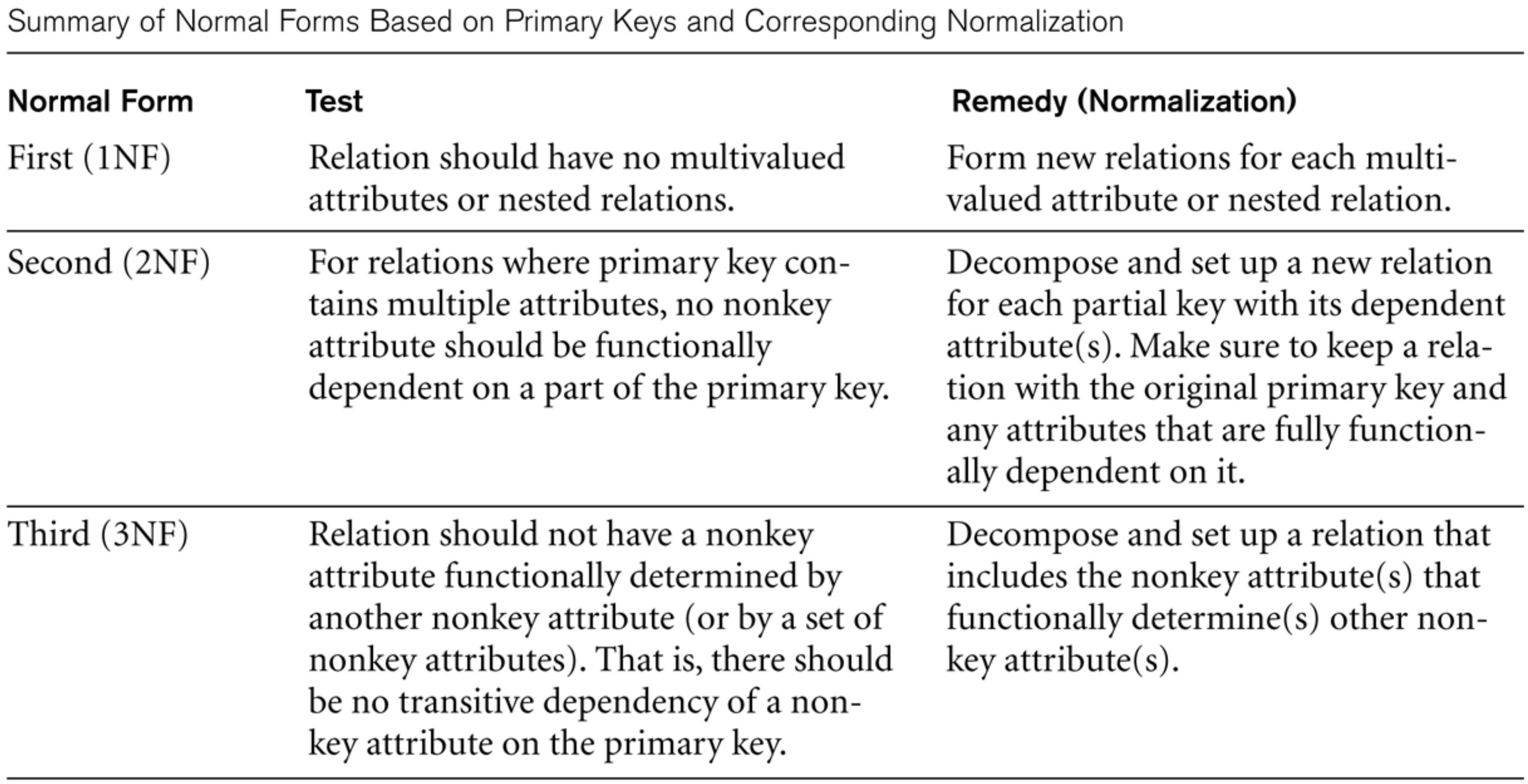
Successive Normalisation of LOTS into 2NF and 3NF



Successive Normalisation of LOTS into 2NF and 3NF



SUMMARY OF NORMAL FORMS based on Primary Keys



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).

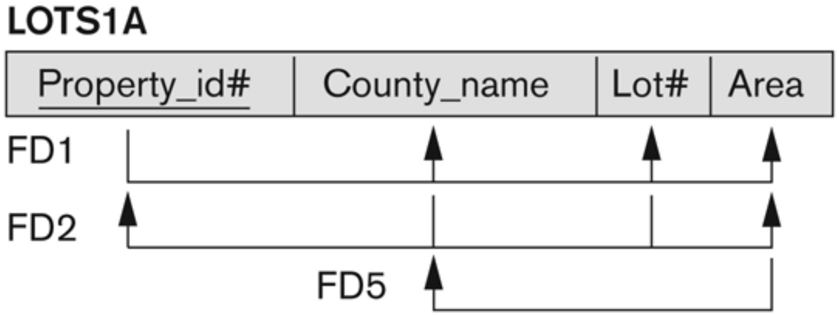
General Normal Form Definitions

- Definition:
  - **Superkey** of relation schema R - a set of attributes S of R that contains a key of R
  - A relation schema R is in **third normal form (3NF)** if whenever a FD X → A holds in R, then either:
    - (a) X is a superkey of R, or
    - (b) A is a prime attribute of R
- NOTE: Boyce-Codd normal form disallows condition (b) above

BCNF (Boyce-Codd Normal Form)

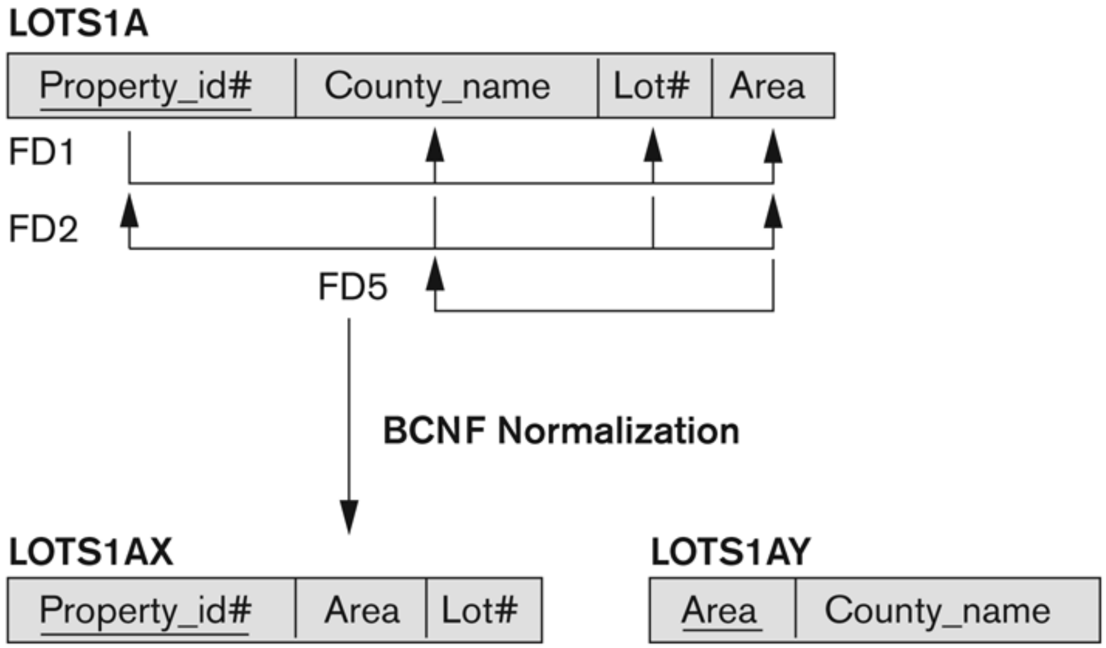
- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever an **FD X → A** holds in R, then **X is a superkey** of R.
- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF
- The goal is to have each relation in BCNF (or 3NF)

Boyce-Codd Normal Form Example



- Suppose that lots in particular counties are of a specific area.
- Then FD Area → County_name holds.
- Is the relation in 3NF? In the relation in BCNF?

Boyce-Codd Normal Form Example



- FD Area → County_name holds.
- The relation is in 3NF but not in BCNF, so decomposition as shown is required.

Boyce-Codd Normal Form

- In practice, most relation schemata that are in 3NF are also in BCNF.
- Only if X → A holds in relation schema R with X *not* being a superkey *and* A being a prime attribute, then R will be in 3NF but NOT in BCNF.

Summary

- Informal Design Guidelines for Relational Databases
- Functional Dependencies (FDs)
- Normal Forms Based on Primary Keys
- General Normal Form Definitions (for multiple keys)
- BCNF (Boyce-Codd Normal Form)