Functional Dependency – Formal Definition

- Suppose that our relational database schema has n attributes: $A_1, A_2, \ldots, A_n$ and the entire database is described by a single, universal relation $R = \{A_1, A_2, \ldots, A_n\}$
  - 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 $t_1[Y] = t_2[Y]$.

Example – Functional Dependencies

- The following FDs may hold, because the four tuples in the current extension have no violation of these constraints.

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

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

- The following FDs do not hold because we already have violations of them in the given extension:

$A \rightarrow B$ [Tuples 1 and 2 violate this constraint]

$B \rightarrow A$ [Tuples 2 and 3 violate this constraint]

$D \rightarrow C$ [Tuples 3 and 4 violate this constraint]

Normalization of Relations

- The normalization process (Codd-1972) takes a relation schema through a series of tests to certify whether it satisfies a certain **normal form.**
  - Top-down process where each relation is evaluated against the criteria for normal forms
  - Relations are decomposed as necessary
- Categories of Normal Forms:
  - First, Second, and Third Normal Forms (1NF, 2NF, 3NF)
  - Boyce-Codd Normal Form (BCNF)
  - Fourth and Fifth Normal Forms (4NF, 5NF) [Theoretical Use]

Normalization of Data

- Process of analyzing relation schemas based on their functional dependencies (FDs) and primary keys to achieve the following properties:
  - Minimizing Redundancy
  - Minimizing insertion, deletion and update anomalies
- "Filtering" or "Purification" process to make relations have successively better quality.
- Relations that don't meet the criteria are decomposed into smaller relation schemas that do meet the criteria.

Normal Forms

- Definition of Normal Form:
  - 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.
- In addition to Normal Forms, we must also check that a relation possesses the following properties:
  - Nonadditive Join (Lossless Join) Property [Most important]
    - Guarantees that spurious tuple generation does not occur
  - Dependency Preservation Property
    - Ensures that each Functional Dependency are preserved during relation decomposition

Types of Keys in a Relation Schema

- A superkey of a relation schema $R = \{A_1, A_2, \ldots, A_n\}$ is a set of attributes $S \subseteq R$ with the property that no two tuples t_1 and t_2 in any legal relation state s of R will have $t_1[S] = t_2[S]$

- A key K is a superkey with the additional property that the removal of any attribute from K will cause K not to be a superkey any more.

Superkeys vs. Keys

- The difference between a key and a superkey is that a key has to be minimal.
  - If we have key $K = \{A_1, A_2, ..., A_k\}$ of R, then $K - \{A_i\}$ is not a key of R for any A_i, $1 \leq i \leq k$.
    - Example:
      - {Ssn} is a key for EMPLOYEE
      - {Ssn}, {Ssn,Ename}, {Ssn,Ename,Bdate} are all superkeys

Keys and Attributes

- 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 secondary keys.
- An attribute of relation 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, since it is not a member of any candidate key.

First Normal Form (1NF)

- Disallows multivalued attributes, composite attributes and their combinations.
- 1NF states that the domain of an attribute must include only atomic (simple, indivisible) values.
- 1NF also states that the value of any attribute in a tuple must be a single value from the domain of that attribute.

1NF - Example

- Consider the DEPARTMENT relation (Pkey:Dnumber)

DEPARTMENT F.K.

Dname	Dnumber	Dmgr_ssn

P.K.

- Suppose that we extend it to include the Dlocations attribute

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations

- We assume that each department can have a number of locations.

DEPARTMENT

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

1NF - Example

- Clear to see that this new DEPARTMENT relation violates First Normal Form *(in fact it isn't even a real relation)*
  - Dlocations is not an atomic attribute
    - The domain of Dlocations contains atomic values, but some tuples can have a set of these values.
    - The domain of Dlocations contains set of values and hence is nonatomic.
- How do we achieve first normal form for such a relation?
  - Three main techniques.

1NF – Example – Technique #1

- Remove the attribute Dlocations that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key Dnumber of DEPARTMENT.
  - The primary key of this new relation is the combination {Dnumber, Dlocation}.
  - A distinct tuple in DEPT_LOCATIONS exists for each location of a department.
  - This decomposes the non-1NF relation into two 1NF relations.

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

1NF – Example – Technique #2

- Expand the key so that there will be a separate table in the original DEPARTMENT relation for each location of a DEPARTMENT.
  - In this case, the primary key becomes the combination {Dnumber, Dlocation}.
  - This solution has the disadvantage of introducing **redundancy** in the relation.

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

1NF – Example – Technique #3

- If the maximum number of values is know for the attribute (say each dept. has a max of 3 locations),
  - Replace the Dlocations attribute by three atomic attributes:
    - Dlocation1, Dlocation2, Dlocation3
  - This solution has the disadvantage of introducing NULL values if most departments have fewer than 3 locations.
    - It also adds a new semantic to the relation about the ordering of the attributes that was not originally intended.
  - Querying becomes more difficult
    - How would we write the following query with this design?
      - List the departments that have 'Bellaire' as one of their locations.

1NF – Resolution Techniques

- The first technique is best since it does not suffer from redundancy and it is completely general.
  - Doesn't limit the maximum number of values

1NF – Nested relations

- 1NF also disallows multivalued attributes that are themselves composite (aka Nested Relations).
  - Each tuple can have a relation within it.
- If the EMP_PROJ relation allowed nesting:

EMP_PROJ		Projs	
Ssn	Ename	Pnumber	Hours

  - Each tuple represents an employee entity, a relation PROJS(Pnumber, Hours) within each tuple represents the employee's projects and the hours per week that employee works on each project.

1NF – Nested Relations

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

1NF – Nested Relations

- This relation violates 1NF. How do we make it 1NF?
  - Notice: Ssn is the primary key and Pnumber is the partial key of the nested relation.
- To normalize this relation to 1NF,:
  - We remove the nested relation attributes into a new relation and propagate the primary key into it.
  - The primary key of the new relation will combine the partial key with the primary key of the original relation.

EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

1NF – Nested Relations

- Decomposition procedure can be applied recursively to a relation with multiple-level nesting to unnest the relation into a set of 1NF relations.
- The existence of more than one multivalued attribute in one relation must be handled carefully.
  - Example non-1NF relation: PERSON (Ss#, {Car_lic#}, {Phone#})
    - A person has multiple cars and multiple phones
    - If we attempt to make all attributes be keys (Technique #2):

      PERSON_IN_1NF (Ss#, Car_lic#, Phone#)

      - Leads to redundancy
    - Decomposition is best (Technique #1)

      P1(Ss#, Car_lic#) and P2(Ss#, Phone#)

Second Normal Form (2NF)

- Based on the concept of full functional dependency
  - A functional dependency X→Y is a full functional dependency if removal of any attribute A from X means that the dependency no longer holds.
  - A functional dependence X→Y is a partial dependency if some attribute A can be removed from X and the dependency still holds.
  - Example:
    - {Ssn, Pnumber} → Hours is a full dependency
      - Since neither Ssn→Hours nor Pnumber→Hours holds
    - {Ssn, Pnumber} → Ename is a partial dependency
      - Since Ssn→Ename holds

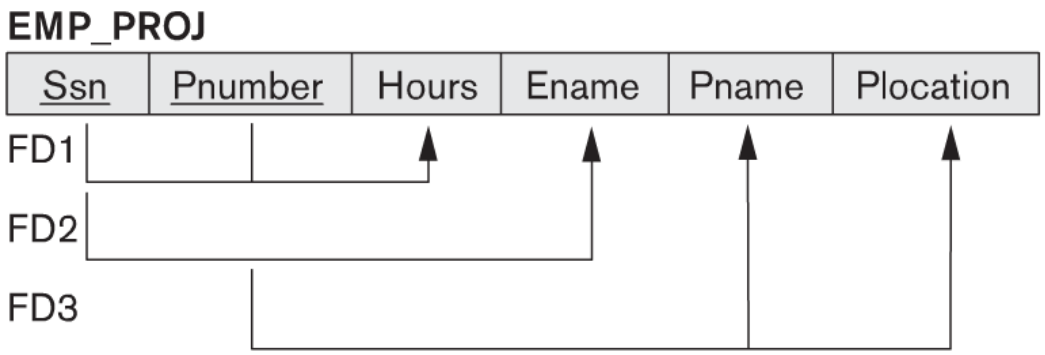
Second Normal Form (2NF)

- 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. $X \rightarrow Y$

- If the primary key contains a single attribute, the test need not be applied at all.

Example



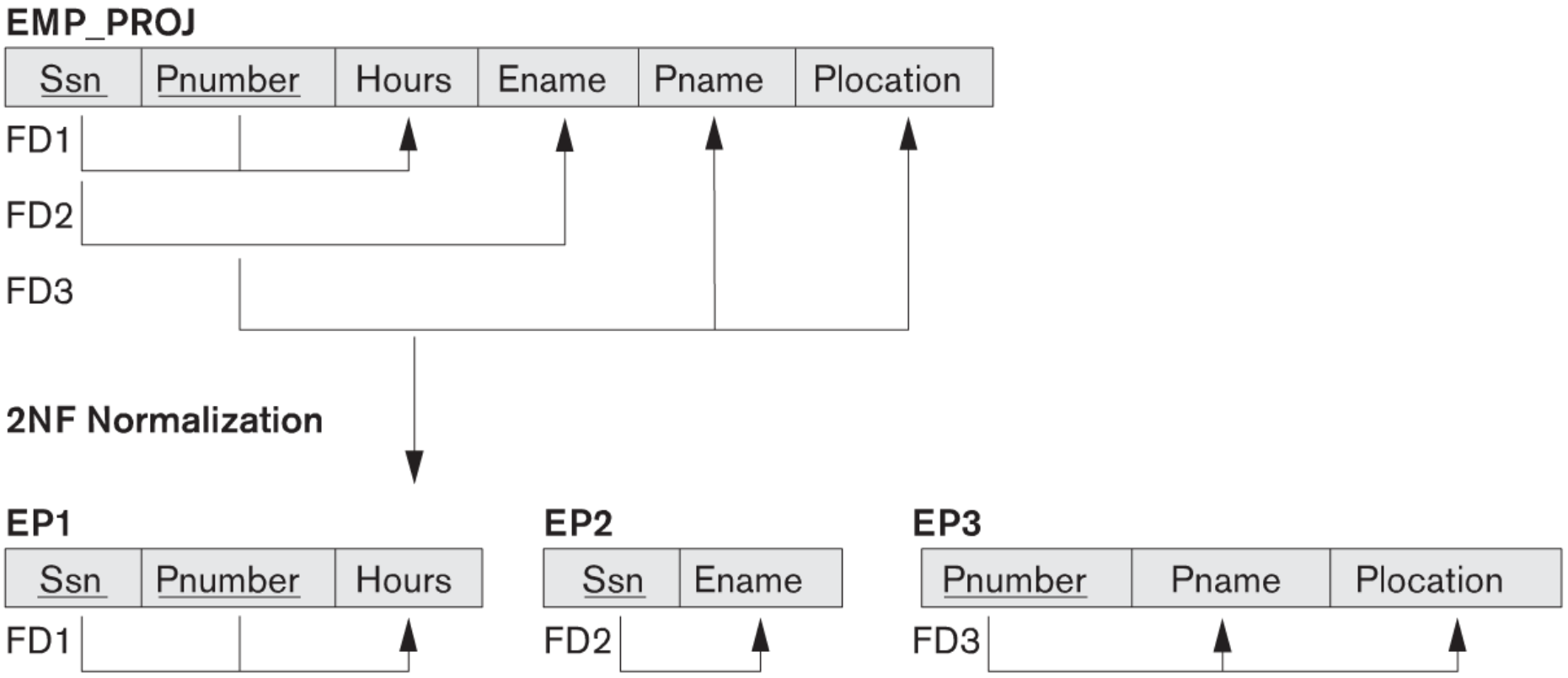
- This relation is in 1NF but not 2NF
  - The non-prime attribute Ename violates 2NF because of FD2
  - Also, the non-prime attributes Pname and Plocation in FD3
- The functional dependencies FD2 and FD3 make Ename, Pname, and Plocation partially dependent on the primary key {Ssn, Pnumber}, thus violating 2NF test.

Normalizing to 2NF

- If a relation schema is not in 2NF:
  - It can be second normalized (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 functionally dependent.

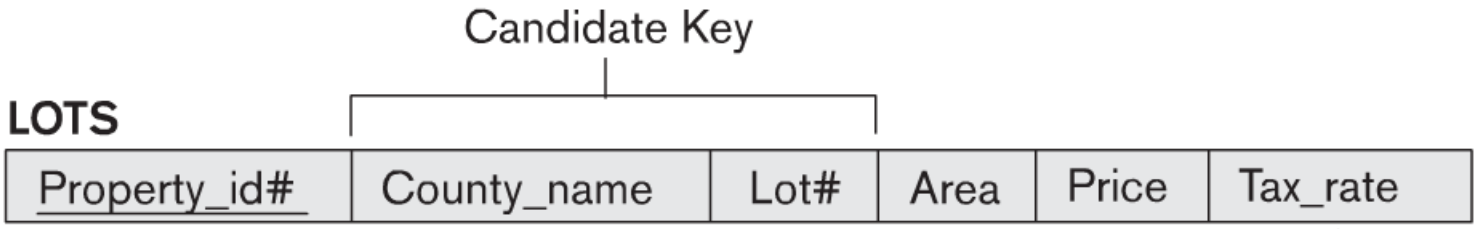
Example

- Normalizing into 2NF:



Example #2 - LOTS

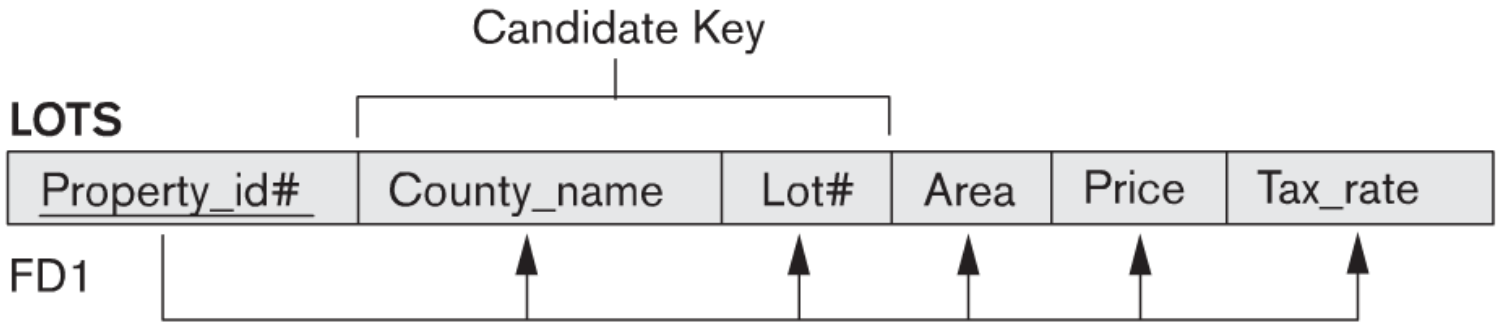
- Consider the following relation schema LOTS, which describes parcels of land for sale in various counties of a state

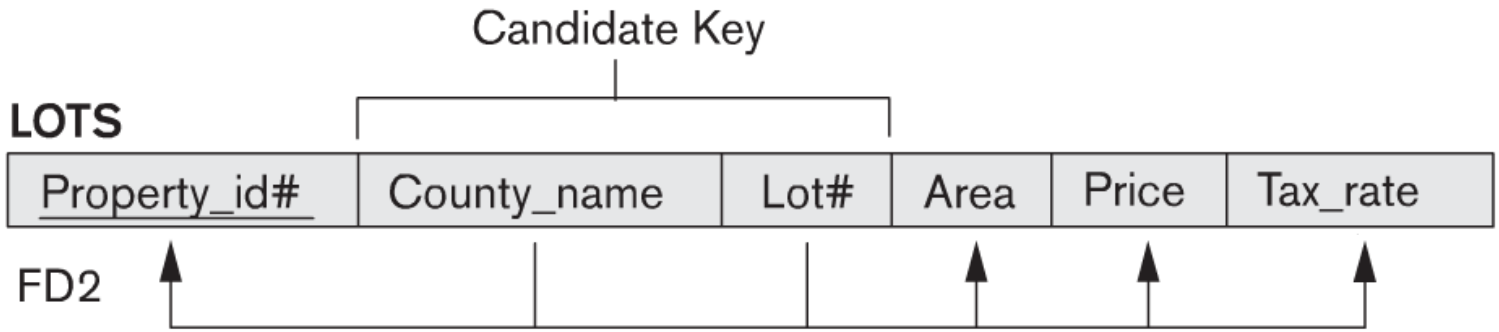


- There are two possible candidate keys: Property_id# and {County_name, Lot#}
  - Lot numbers are unique only within each county
  - Property_id# numbers are unique across counties for the state

Example #2 - LOTS

- Based on the two candidate keys, FD1 and FD2 hold.



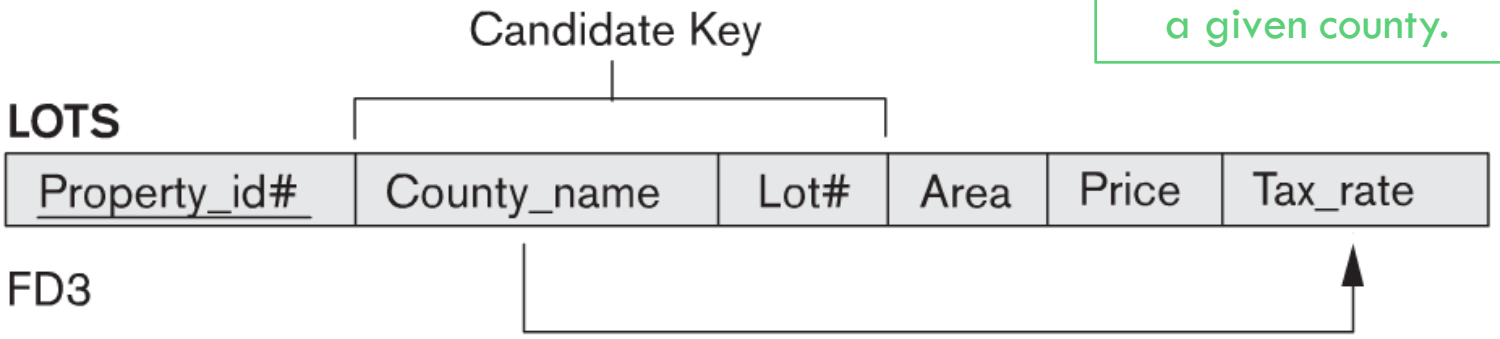


- Suppose we choose Property_id# to be the primary key.

Example #2 - LOTS

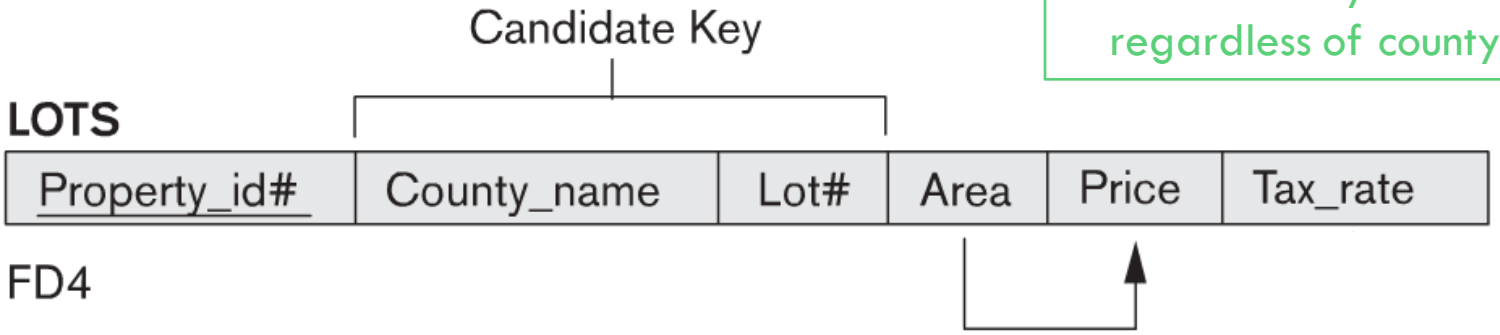
- Suppose that the following two FDs also hold in LOTS:
  - FD3: County_name → Tax_rate

Tax Rate is fixed for a given county.



  - FD4: Area → Price

The price of a lot is determined by its area regardless of county.



Example #2 - LOTS

- LOTS violates the general definition of 2NF:
  - Tax_rate is partially dependent on the candidate key {County_name, Lot#} due to FD3.
- To normalize LOTS into 2NF, we use decomposition:
  - Decompose into two relations LOTS1 and LOTS2

LOTS1

Property_id#	County_name	Lot#	Area	Price

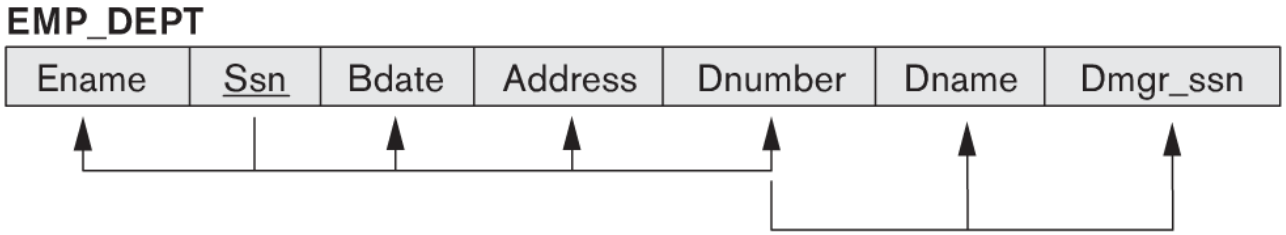
LOTS2

County_name	Tax_rate

  - Construct LOTS1 by removing Tax_rate that violates 2NF and place it with County_name (left-hand side of FD3 that causes partial dependency) into another relation LOTS2.
  - Both LOTS1 and LOTS2 are now 2NF.
  - Note: FD4 does not violate 2NF and is carried over to LOTS1.

Third Normal Form (3NF)

- Third normal form (3NF) is based on the concept of transitive dependency.
  - A functional dependency X→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→Z and Z→Y hold.
  - Example:
    - The dependency Ssn→Dmgr_ssn is transitive through Dnumber in EMP_DEPT. (Notice: Dnumber is not a key or a subset of a key.)

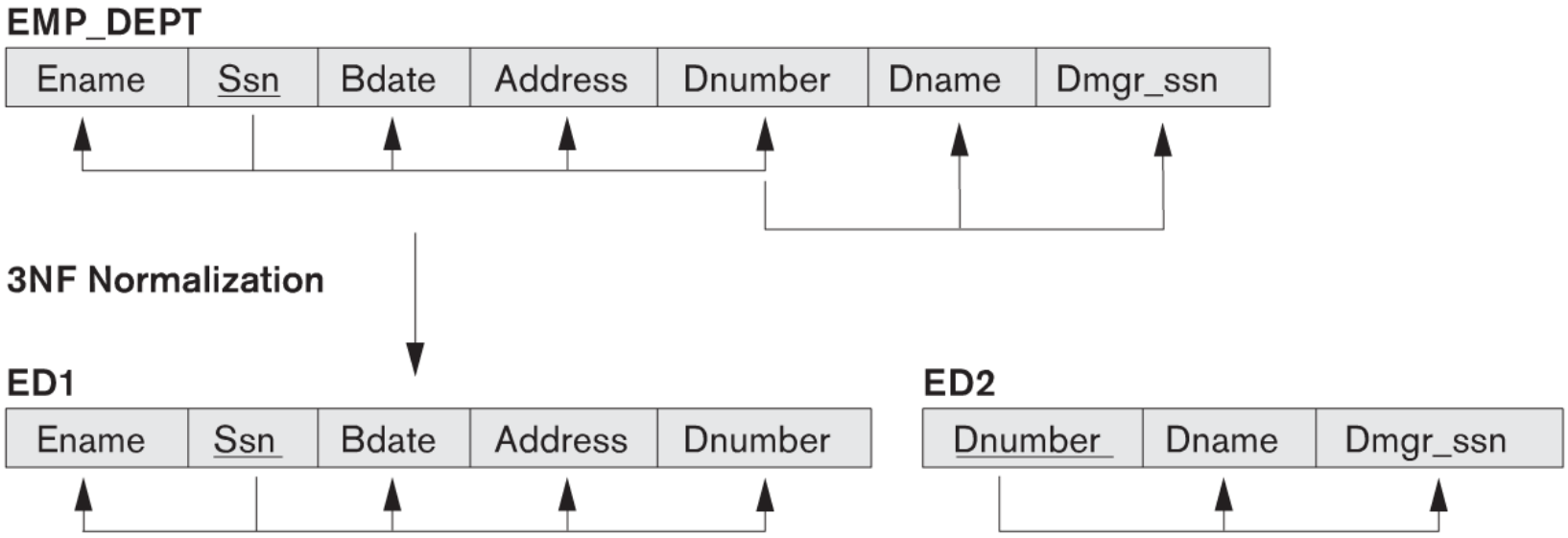


Third Normal Form (3NF)

- 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.
    - EMP_DEPT relation is in 2NF, since no partial dependencies on a key exist.
    - EMP_DEPT relation is not in 3NF because of the transitive dependency of Dmgr_ssn (and also Dname) on Ssn via Dnumber.
- General Definition:
  - A relation schema R is in 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.

Normalizing into 3NF

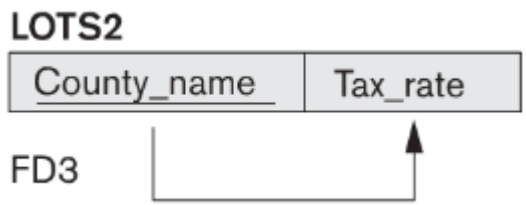
- Normalize EMP_DEPT by decomposing it into two 3NF relation schemas ED1 and ED2.



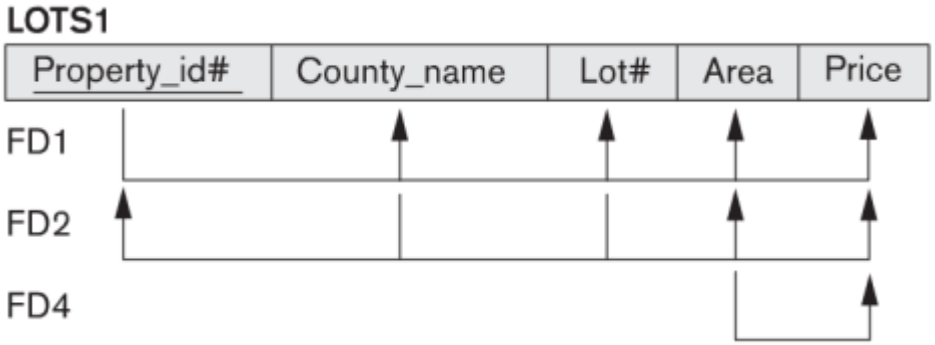
- ED1 and ED2 represent independent entity facts about employees and departments.
  - Natural Join will recover EMP_DEPT w/o spurious tuples.

Example #2 - LOTS

- According the 3NF definition:
  - LOTS2 is in 3NF



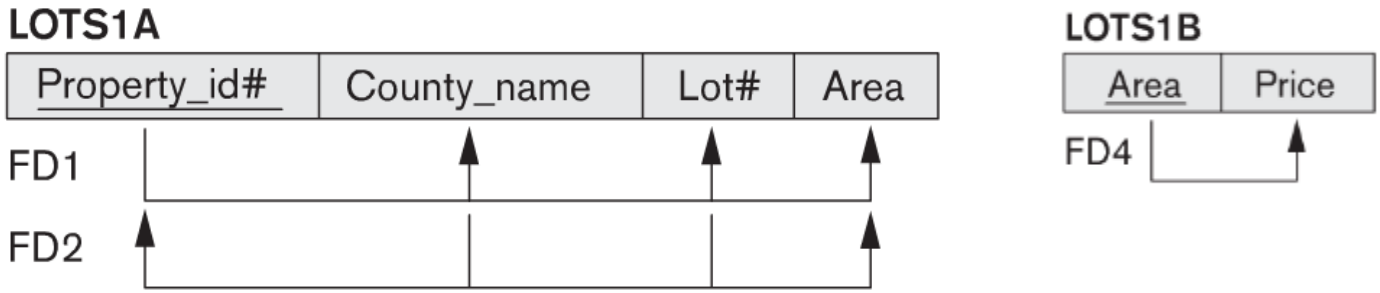
  - LOTS1 is NOT in 3NF
    - FD4: Area→Price
    - Area is not a superkey



    - Price is transitively dependent on each of the candidate keys of LOTS1 via the nonprime attribute Area.

Example #2 - LOTS

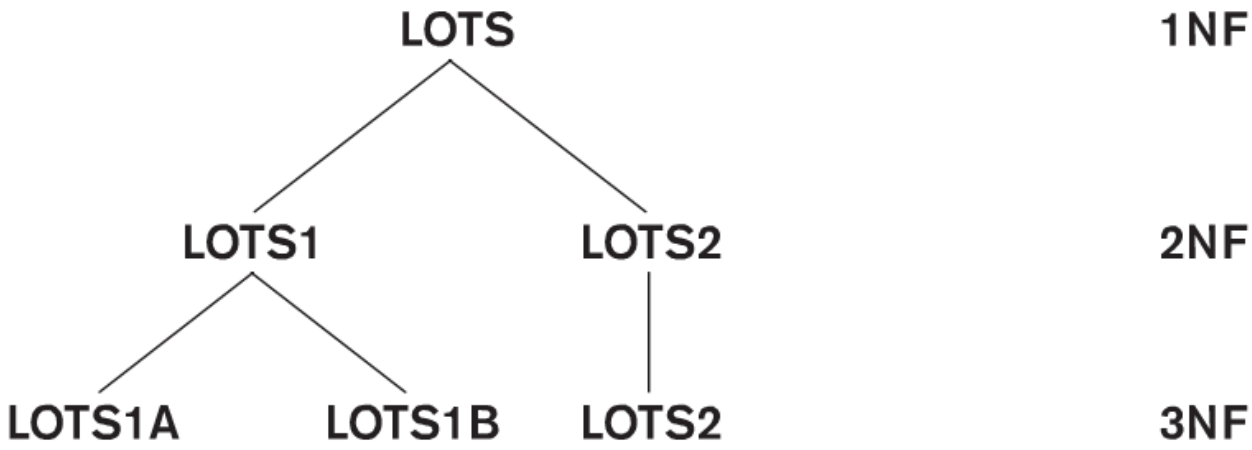
- Decomposition of LOTS1 into LOTS1A and LOTS1B:



- Construct LOTS1A by removing the attribute Price that violates 3NF from LOTS1 and placing it with area (left-hand side of FD4 that causes the transitive dependency) into another relation LOTS1B.

- Both LOTS1A and LOTS1B are both in 3NF.

Example #2 - Summary



- Note: We could have decomposed LOTS directly into LOTS1A, LOTS1B and LOTS2.
- The transitive and partial dependencies that violate 3NF can be removed in any order.

Third Normal Form (3NF)

- Alternative definition for 3NF:
  - 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.

Summary of 1NF, 2NF and 3NF

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).