CS322: Database Systems

Normalization

Dr. Manas Khatua
Assistant Professor
Dept. of CSE
IIT Jodhpur
E-mail: manaskhatua@iitj.ac.in

Introduction

- The normalization process
  - takes a relation schema through a series of tests to *certify* whether it satisfies a certain normal form.
  - otherwise, decompose relations as necessary.

- The normalization process provides database designers with the following:

  - A formal framework for analyzing relation schemas based on their keys and on the FDs 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

- Normalized to a desired degree for:
  - minimizing redundancy
  - minimizing the insertion, deletion, and update anomalies

- It is considered as *relational design by analysis.*

Normal Forms

- Normal Forms:
  - Based on FDs among the attributes of a relation
    - first normal form (1NF)
    - second normal form (2NF)
    - third normal form (3NF)
    - Boyce-Codd normal form (BCNF)
  - Based on multivalued FDs
    - fourth normal form (4NF)
  - Based on join FDs
    - fifth normal form (5NF)
- Normal form of a relation:
  - 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.
- Denormalization:
  - It is the process of storing the join of higher normal form relations as a base relation, which is in a lower normal form.

Decomposition Rules

- Normal forms, when considered *in isolation* from other factors, do not guarantee a good database design.

- the process of normalization through decomposition must also confirms the following:

  - The nonadditive join or lossless join property, which guarantees that the spurious tuple generation problem does not occur w.r.t. the relation schemas created after decomposition.

  - The dependency preservation property, which ensures that each FD is represented in some individual relation resulting after decomposition.

- The nonadditive join property is extremely critical and *must be achieved at any cost*,
- whereas the dependency preservation property, although desirable, is sometimes sacrificed.

A Lossy Decomposition

ID	*name*	*street*	*city*	*salary*
⋮				
57766	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

employee

ID	*name*
⋮	
57766	Kim
98776	Kim
⋮	

name	*street*	*city*	*salary*
⋮			
Kim	Main	Perryridge	75000
Kim	North	Hampton	67000
⋮			

natural join

ID	*name*	*street*	*city*	*salary*
⋮				
57766	Kim	Main	Perryridge	75000
57766	Kim	North	Hampton	67000
98776	Kim	Main	Perryridge	75000
98776	Kim	North	Hampton	67000
⋮				

It occurs when we don't follow the Primary Key – Foreign Key relationship.

Many spurious tuples are generated.

Example of Lossless-Join Decomposition

- **Lossless join decomposition**

- Decomposition of $R = (A, B, C)$ into $R_1 = (A, B)$ $R_2 = (B, C)$

A	B	C
α	1	A
β	2	B

r

A	B
α	1
β	2

$\Pi_{A,B}(r)$

B	C
1	A
2	B

$\Pi_{B,C}(r)$

$\Pi_A (r) \bowtie \Pi_B (r)$

A	B	C
α	1	A
β	2	B

Dependency Preservation

- Let F_i be the set of dependencies in F^+ that include only attributes in relation R_i.
  - A decomposition is **dependency preserving**, if
    $(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$

- *Example:*
  $R = (A, B, C)$
  $F = \{A \rightarrow B$
  $B \rightarrow C\}$

  Candidate Key = $\{A\}$

  Decomposition
  $R_1 = (A, B),\ R_2 = (B, C)$

  Here, FDs are preserved.

- *Example:*
  $R = (A, B, C)$
  $F = \{AB \rightarrow C$
  $C \rightarrow B\}$
  Candidate Keys = $\{AB\}$, {AC}

Decomposition
$R_1 = (A, B),\ R_2 = (B, C)$
OR, $R_1 = (A, C),\ R_2 = (B, C)$

Here, FDs are **not** preserved.

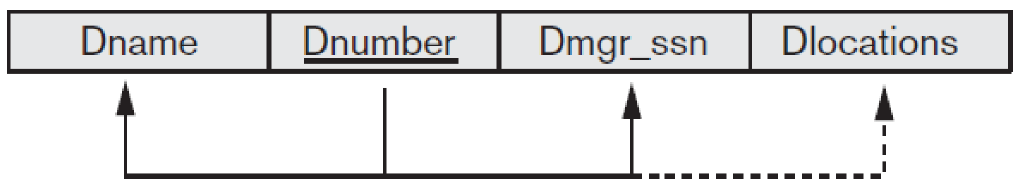
First Normal Form (1NF)

- A relational schema R is in **first normal form** (1NF) if the domains of all attributes of R are atomic.

- A domain is **atomic** if its elements are considered to be indivisible units.
- i.e., it disallow multivalued attributes, composite attributes, and their combinations.

- Non-atomic values complicate storage and encourage redundant storage of data

DEPARTMENT



DEPARTMENT

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

Multivalued Attribute

Cont…

EMP_PROJ

		Projs	
Ssn	Ename	Pnumber	Hours

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

Composite Attribute

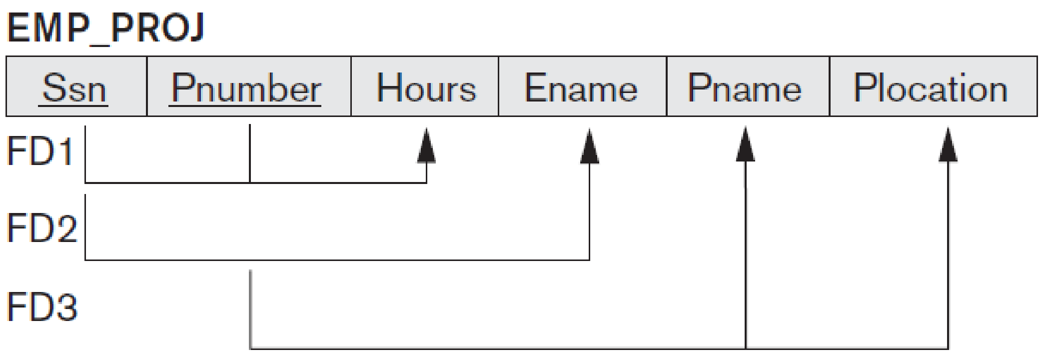
- Few techniques for decomposing to 1NF:
  1. Remove the attribute that violates 1NF and place it in a separate relation
  2. If a *maximum number of values* is known for the attribute, then replace the attribute by *k* atomic attributes such as *Dlocation1, Dlocation2, …, Dlocationk.*
  3. *Few more …*

- the first is generally considered best because
  - it does not suffer from redundancy and
  - it is completely general (no need to know the limits)

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.

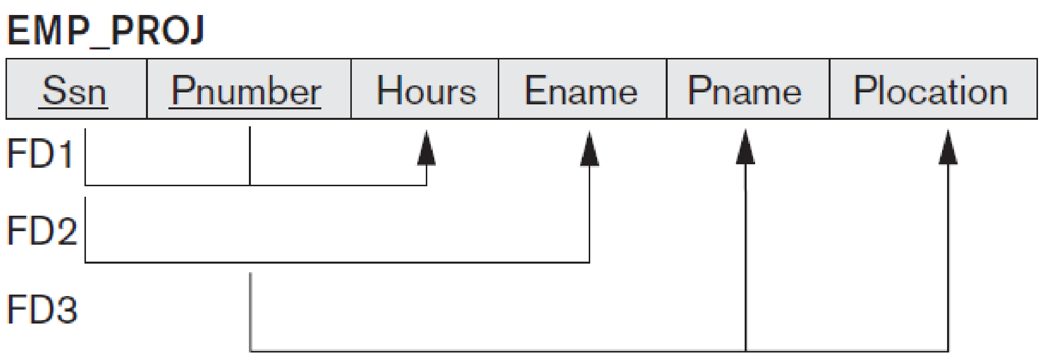
- A functional dependency $X \rightarrow Y$ is a **partial dependency** if some attribute $A \in X$ can be removed from X and the dependency still holds; that is, for some $A \in X$, $(X - \{A\}) \rightarrow Y$.



- {Ssn, Pnumber} → Hours is a full dependency (neither Ssn → Hours nor Pnumber→Hours holds).
- However, the dependency {Ssn, Pnumber}→Ename is partial because Ssn→Ename holds.

Second Normal Form (2NF)

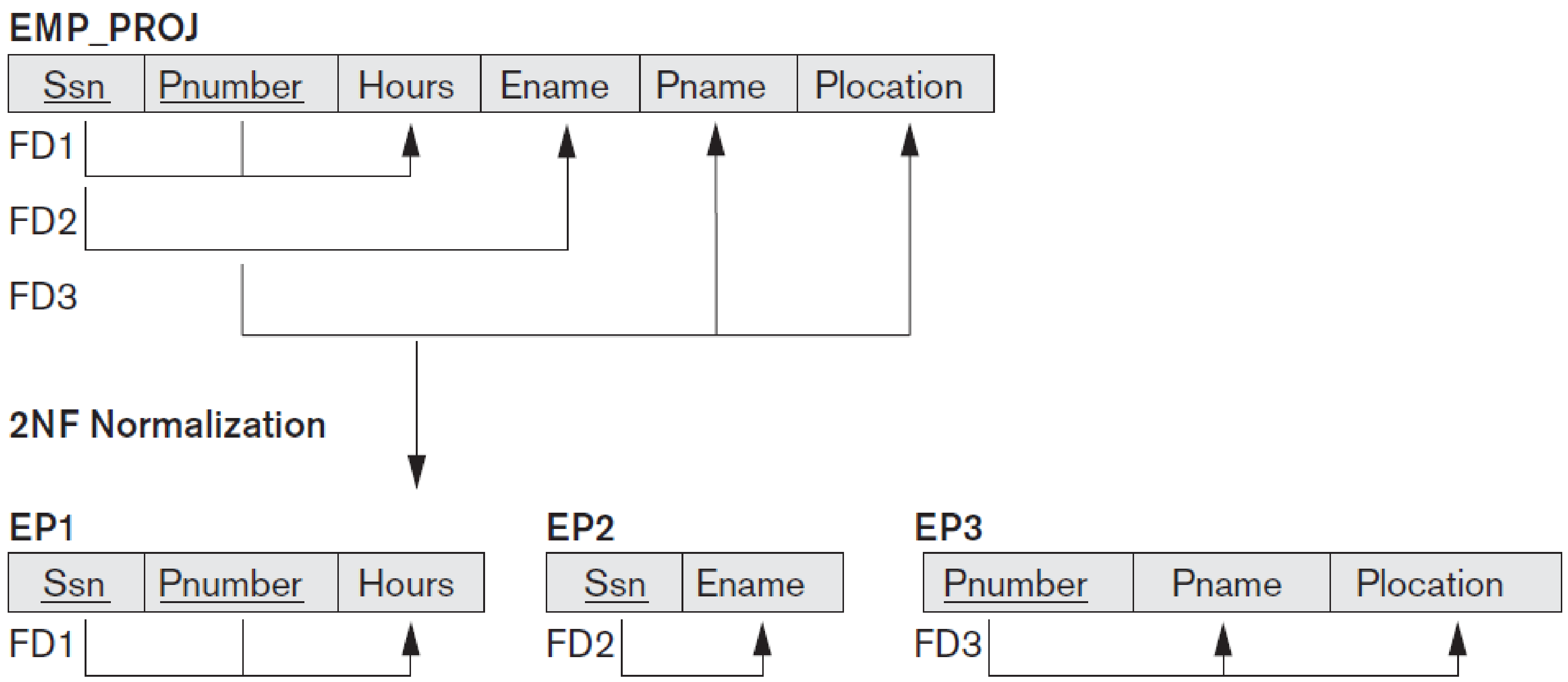
- A relation schema *R* is in 2NF if
  - *R* is in 1NF, and
  - every nonprime attribute *A* in *R* is *fully functionally dependent* on the primary key of *R*.



- The EMP_PROJ relation is in 1NF but is not in 2NF.
- The nonprime attribute *Ename* violates 2NF.

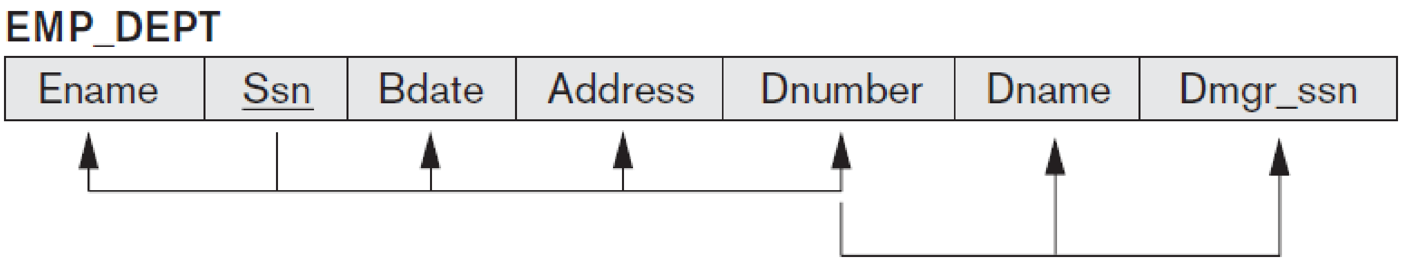
Cont…

- It can be *2NF normalized* if it is decomposed to the relations in which nonprime attributes are associated only with the part of the primary key on which they are fully functionally dependent.



Transitive Functional 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.

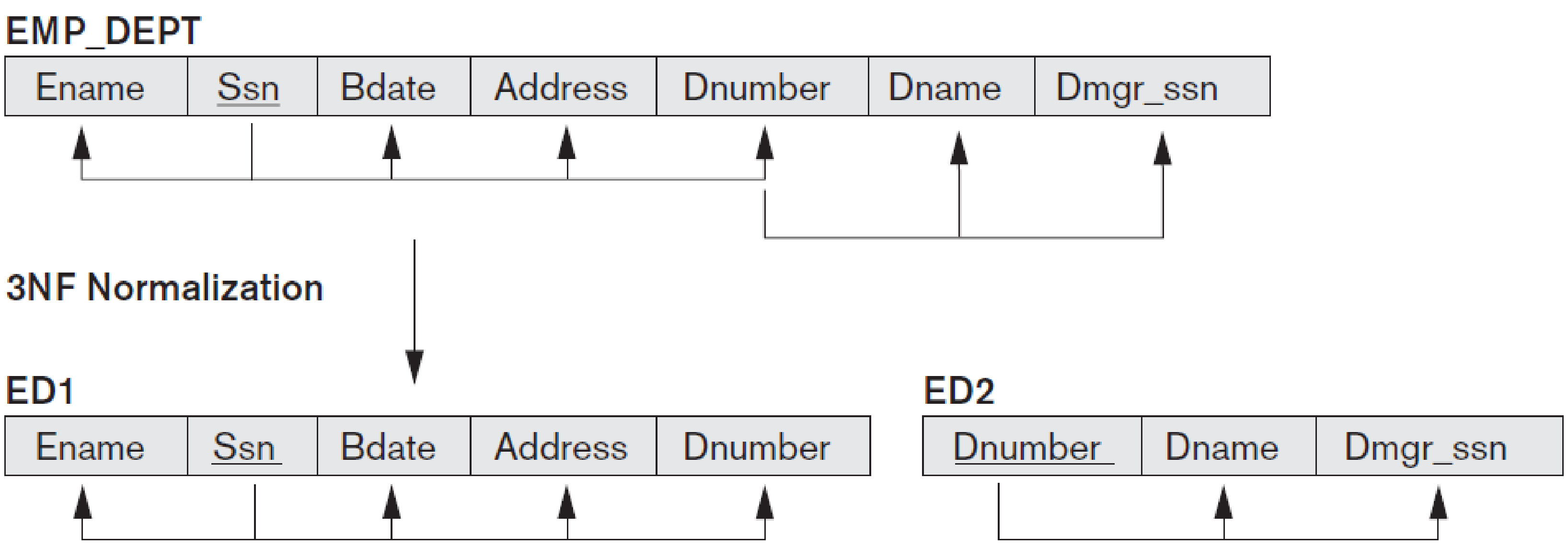


- the dependencies {Ssn → Dnumber} and {Dnumber → Dmgr_ssn} hold *and Dnumber* is neither a key itself nor a subset of the key of EMP_DEPT.

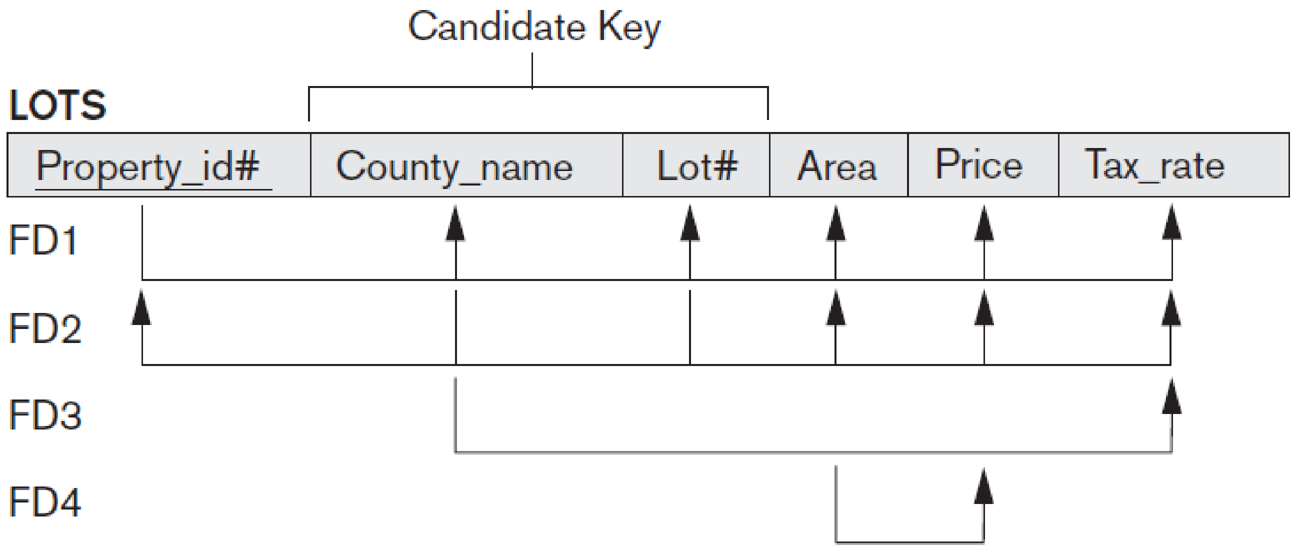
Third Normal Form (3NF)

- A relation schema *R* is in **3NF** if
  - it satisfies 2NF *and*
  - no nonprime attribute of *R* is transitively dependent on the primary key.

- Solution:
  - Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).



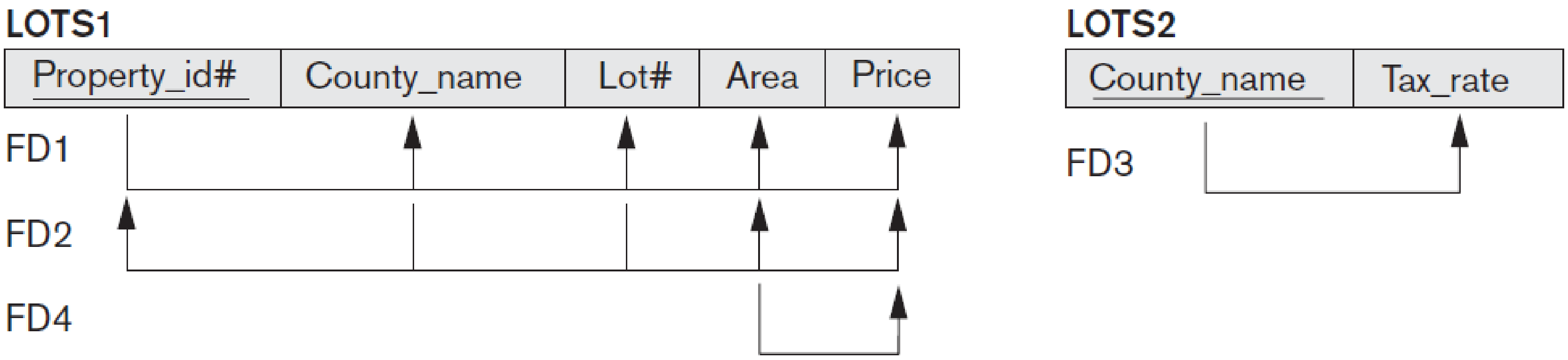
Example (1NF-3NF)



- The *LOTS* relation is in 1NF as the domains of all attributes are atomic.
- It is not in 2NF as the FD3 is not fully functionally dependent on candidate key.
- So, it is not in 3NF as well.

Cont...

- Normalize to 2NF

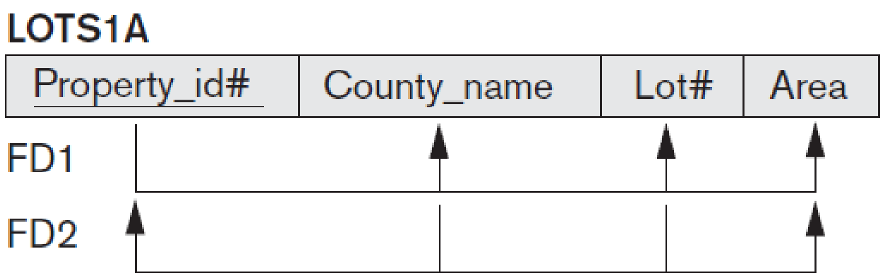


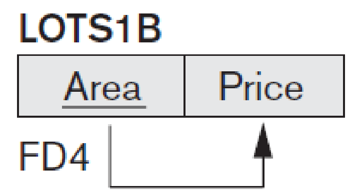
- LOTS1 is in 2NF, but not in 3NF as FD4 is a transitive functional dependency.

- LOTS2 is in 3NF.

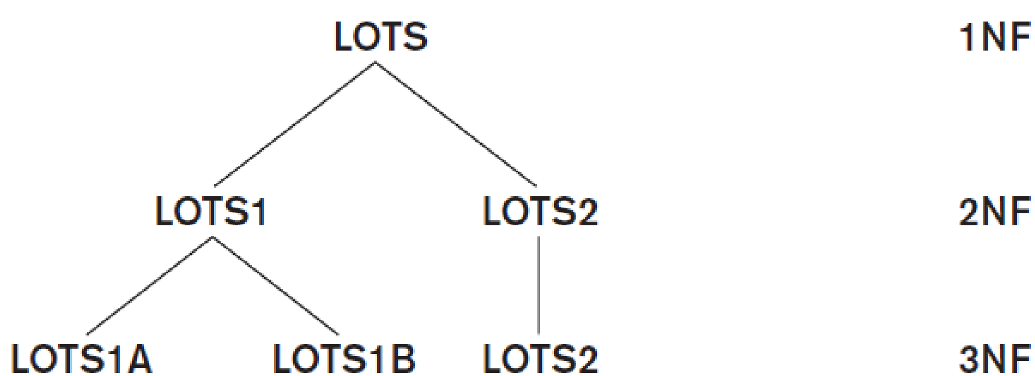
Cont...

- Normalize to 3NF





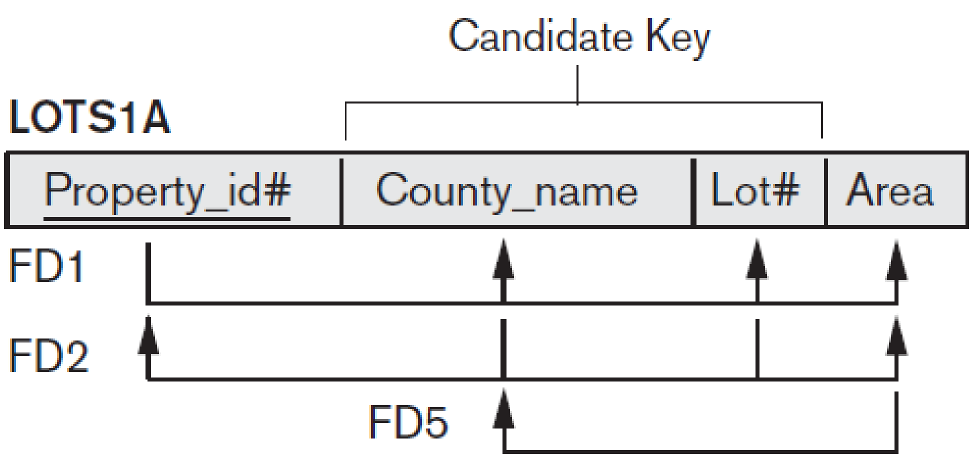
- LOST1A is in 3NF
- LOST1B is in 3NF
- Finally, we get



Boyce-Codd Normal Form (BCNF)

- *A* functional dependency $\alpha \rightarrow \beta$ is **trivial** if $\beta \subseteq \alpha$
  - Example*:*
    - *ID, name* $\rightarrow$ *ID*
    - *name* $\rightarrow$ *name*

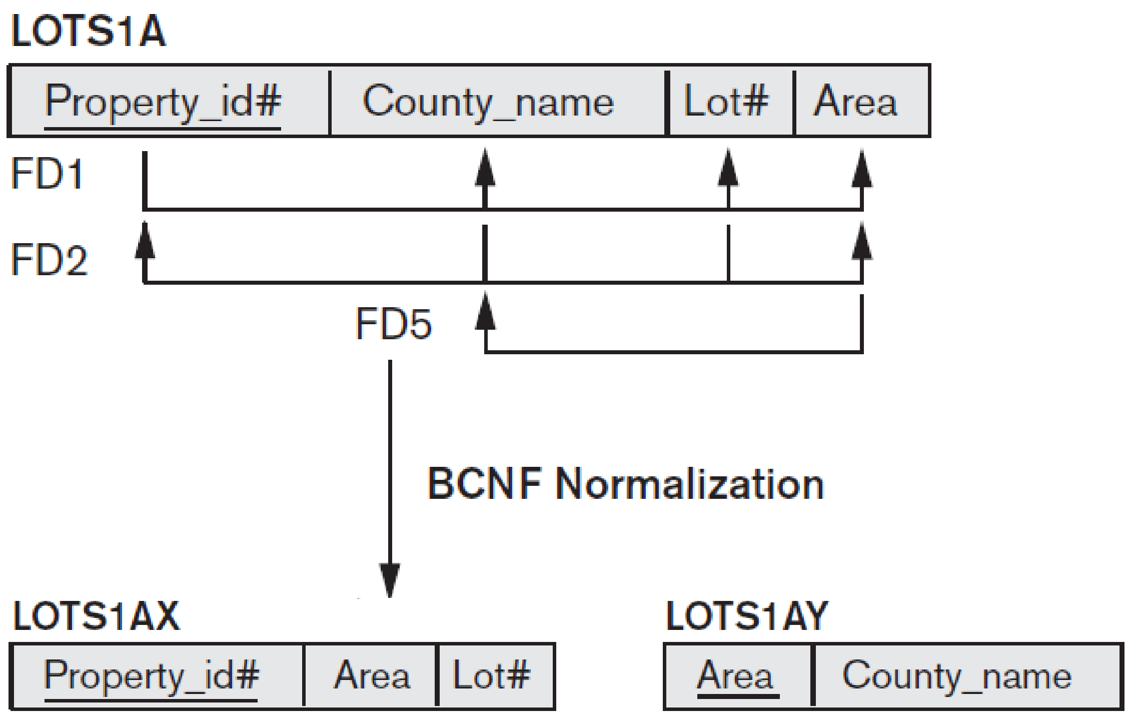
- A relation schema *R* is in **BCNF** if
  - the relation R is in 3NF, and
  - whenever a *nontrivial* functional dependency *X*→*A* holds in *R*, then *X* is a superkey of *R*.



- LOTS1A is in 3NF, but not in BCNF as the attribute Area in FD5 is not a superkey of LOTS1A.

Cont…

- Normalization to BCNF



Example (in 3NF but not in BCNF)

- *R = (A, B, C)*

  $F = \{\ AB \rightarrow C;\ C \rightarrow B\ \}$

Then, R is in 3NF, but not in BCNF.

- Let a relation TEACH

FD1: {Student, Course} → Instructor

FD2: Instructor → Course

Candidate Key: {Student, Course}

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

- It may be decomposed into one of the three following possible pairs:
  - R1. {Student, Instructor} and {Student, Course}
  - R2. {Course, Instructor} and {Course, Student}
  - R3. {Instructor, Course} and {Instructor, Student}
- All three decompositions *lose the functional dependency* FD1.
- But, nonadditive decomposition is a must during normalization. So, let us test those relations:

- R1 and R2 do not satisfy nonadditive decomposition, but R3 does.
- Hence, R3 is the correct decomposition

Comparison of BCNF and 3NF

- It is always possible to decompose a relation into a set of relations that are in 3NF such that:
  - the decomposition is lossless
  - the dependencies are preserved

- It is always possible to decompose a relation into a set of relations that are in BCNF such that:
  - the decomposition is lossless
  - it may not be possible to preserve dependencies.

Design Goals

- Primary goal for a relational database design is to achieve:
  - BCNF.
  - Lossless join.
  - Dependency preservation.

- If we cannot achieve this, we accept one of
  - Lack of dependency preservation
  - Redundancy due to use of 3NF

- Interestingly, SQL does not provide a direct way of specifying FDs other than superkeys.

  Can specify FDs using assertions, but they are expensive to test, (and currently not supported by any of the widely used databases!)

- Even if we had a dependency preserving decomposition, using SQL we would not be able to efficiently test a functional dependency whose left hand side is not a key.

Multivalued Dependency (MVD)

- Let R be a relation schema, and let $\alpha \subseteq R$ and $\beta \subseteq R$.
- The **multivalued dependency**

$$\alpha \rightarrow\rightarrow \beta$$

holds on R for any legal relation $r(R)$: if two tuples t_1 and t_2 exists in r such that $t_1[\alpha] = t_2[\alpha]$, then there exist two tuples t_3 and t_4 in r such that:

$$t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$$

$$t_3[\beta] = t_1[\beta]$$
$$t_3[R - \alpha - \beta] = t_2[R - \alpha - \beta]$$

$$t_4[\beta] = t_2[\beta]$$
$$t_4[R - \alpha - \beta] = t_1[R - \alpha - \beta]$$

- Tabular representation of $\alpha \rightarrow\rightarrow \beta$

	α	β	$R - \alpha - \beta$
t_1	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$a_{j+1} \ldots a_n$
t_2	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$b_{j+1} \ldots b_n$
t_3	$a_1 \ldots a_i$	$a_{i+1} \ldots a_j$	$b_{j+1} \ldots b_n$
t_4	$a_1 \ldots a_i$	$b_{i+1} \ldots b_j$	$a_{j+1} \ldots a_n$

Example

- Suppose we record names of children, and phone numbers for instructors:
  - *inst_child* (*Inst_ID*, *child_name*)
  - *inst_phone* (*Inst_ID*, *phone_number*)

- If we were to combine these schemas to get
  - *inst_info*(*Inst_ID*, *child_name*, *phone_number*)

  - Example data:
    (99999, David, 512-555-1234) ← t1
    (99999, David, 512-555-4321) ← t3
    (99999, William, 512-555-1234) ← t2
    (99999, William, 512-555-4321) ← t4

Conditions: $t_1[\alpha] = t_2[\alpha] = t_3[\alpha] = t_4[\alpha]$

$$t_3[\beta] = t_1[\beta]$$
$$t_3[R - \alpha - \beta] = t_2[R - \alpha - \beta]$$

$$t_4[\beta] = t_2[\beta]$$
$$t_4[R - \alpha - \beta] = t_1[R - \alpha - \beta]$$

Fourth Normal Form (4NF)

- A relation schema *R* is in **4NF** with respect to a set *D* of functional and multivalued dependencies if for all multivalued dependencies in D^+ of the form $\alpha \rightarrow\rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following hold:
  - $\alpha \rightarrow\rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$ or $\alpha \cup \beta = R$)
  - α is a superkey for schema *R*

- If a relation is in 4NF it is in BCNF

Example

- Normalize to 4NF

- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow\rightarrow B$
  $B \rightarrow\rightarrow HI$
  $CG \rightarrow\rightarrow H \}$
- R is not in 4NF since $A \rightarrow\rightarrow B$ and A is not a superkey for R

- Decomposition
  a) $R_1 = (A, B)$ (R_1 is in 4NF)
  b) $R_2 = (A, C, G, H, I)$ (R_2 is not in 4NF, decompose into R_3 and R_4)

  c) $R_3 = (C, G, H)$ (R_3 is in 4NF)
  d) $R_4 = (A, C, G, I)$ (R_4 is not in 4NF, decompose into R_5 and R_6)
  - $A \rightarrow\rightarrow B$ and $B \rightarrow\rightarrow HI$ ➔ $A \rightarrow\rightarrow HI$, (MVD transitivity), and
  - and hence $A \rightarrow\rightarrow I$ *(MVD restriction to R_4)*

  e) $R_5 = (A, I)$ (R_5 is in 4NF)
  f) $R_6 = (A, C, G)$ (R_6 is in 4NF)

Example

- The EMP relation with two MVDs:
  Ename →→ Pname and Ename →→ Dname

EMP

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

- Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

EMP_PROJECTS

Ename	Pname
Smith	X
Smith	Y

EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

Thanks!