NORMALIZATION

- Normalization is a process of analyzing the given relation schemas based on their Functional Dependencies and primary keys to achieve the desirable properties of

  (1) Minimizing redundancy and

  (2) Minimizing the insertion, deletion, and update anomalies

Normalization of Relations:

- The normalization process, as first proposed by Codd (l972), takes a relation schema through a series of tests to "certify" whether it satisfies a certain normal form.
- Codd proposed three normal forms: 1NF, 2NF, and 3NF.
- The process proceeds in a top-down fashion by evaluating each relation against the criteria for normal forms and decomposing relations as necessary. It is also called as relational design by analysis.
- Thus, the normalization procedure provides database designers with the following:

  i) A formal framework for analyzing relation schemas based on their keys and on the functional dependencies among their attributes.

  ii) 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.
- 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.

- The process of normalization through decomposition must also confirm the existence of additional properties that the relational schemas should possess. These would include two properties:

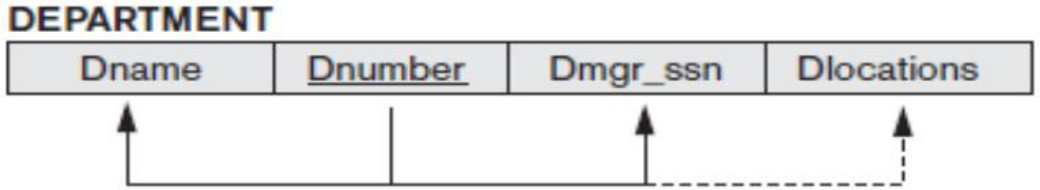
  a) The lossless join or nonadditive join property: This guarantees that the spurious tuple generation problem does not occur with respect to the relation schemas created after decomposition.

  b) The dependency preservation property: This ensures that each functional dependency is represented in some individual relation resulting after decomposition.

- The process of storing the join of higher normal form relations as a base relation-which is in a lower normal form-is known as “denormalization”. This is sometimes done for some performance reasons.

First Normal Form:

- First normal form (INF) is defined to disallow **multivalued attributes, composite attributes, and their combinations.**
- 1NF states that “the domain of an attribute must include only **atomic** (simple, indivisible) values and that the value of any attribute in a tuple must be a **single value** from the domain of that attribute”.
- Consider the DEPARTMENT relation schema shown whose primary key is DNUMBER, and suppose that we extend it by including the DLOCATIONS attribute as shown . We assume that each department can have a number of locations. The example relation state for DEPARTMENT .



DEPARTMENT

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

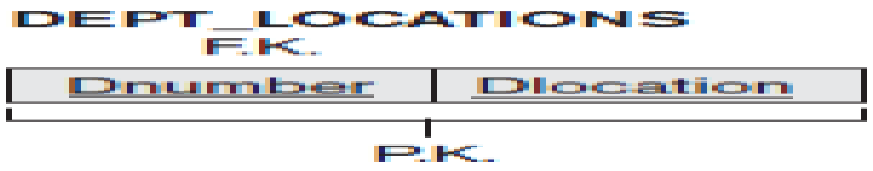
There are three main techniques to achieve first normal form for such a relation:

- a) Remove the attribute DLOCATIONS that violates 1NF and place it in a separate relation DEPT_LOCATIONS along with the primary key DNUMBER DEPARTMENT.

i) The primary key of this relation is the combination {DNUMBER, DLOCATION}, as shown in following Figure.

ii) A distinct tuple in DEPT_LOCATIONS exists for each location of a department.

iii) This decomposes the non-1NF relation into two 1NFrelations



- b) Expand the key so that there will be a separate tuple in the original DEPARTMENT relation for each location of a DEPARTMENT, as shown in the following

  i) In this case, the primary key becomes the combination {DNUMBER, DLOCATION}.

  ii) 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

c) If a maximum number of values is known for the attribute-for example, if it is known that at most three locations can exist for a department-replace the DLOCATIONS attribute by three atomic attributes: DLOCATIONl, DLOCATION2, and DLOCATION3.

i)This solution has the disadvantage of introducing null values if most departments have fewer than three locations.

ii) It further introduces a spurious semantics about the ordering among the location values that is not originally intended.

- Of the three solutions above, the first is generally considered best because it does not suffer from redundancy and it is completely general, having no limit placed on a maximum number of values.

Example

The EMP_PROJ relation schema could appear if nesting is allowed. Each tuple Represents an employee entity, and a relation PROJS(PNUMBER, HOURS) within each tuple represents the employee's projects and the hours per week that employee works on each project. The schema of this EMP_PROJ relation can be represented as follows:

EMP_PROJ (SSN, ENAME, {PROJS(PNUMBER, HOURS)})

- The set braces { } identify the attribute PROJS as multivalued, and we list the component attributes that form PROJS between parentheses ( )

(a)
EMP_PROJ

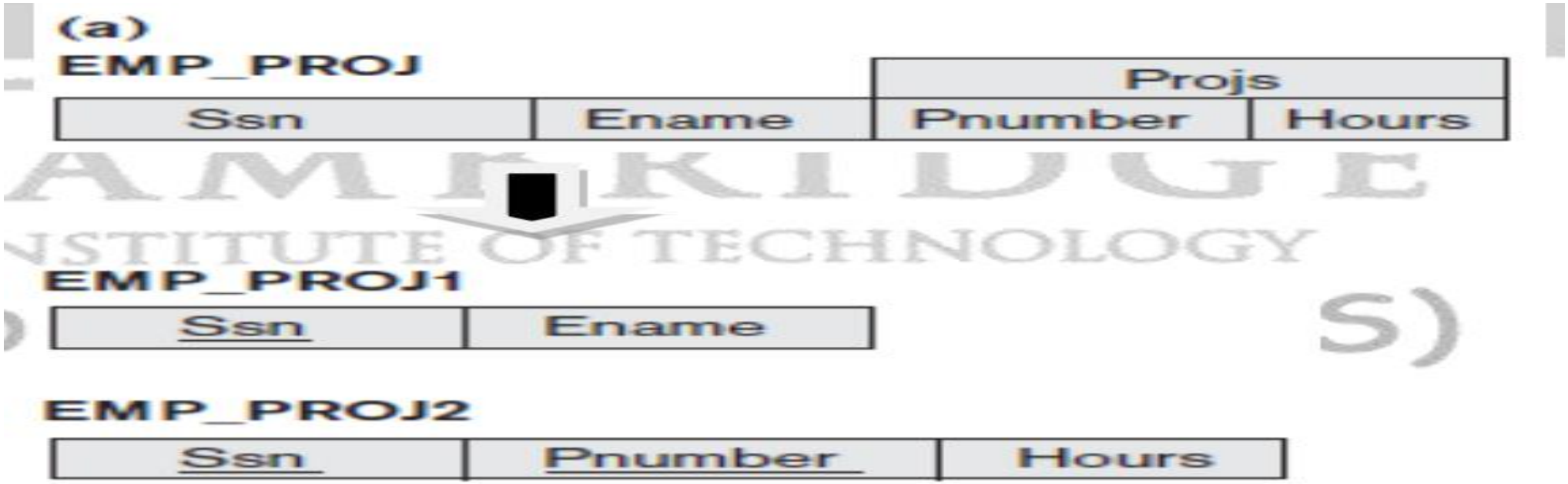
		Projs	
Ssn	Ename	Pnumber	Hours

(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, 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

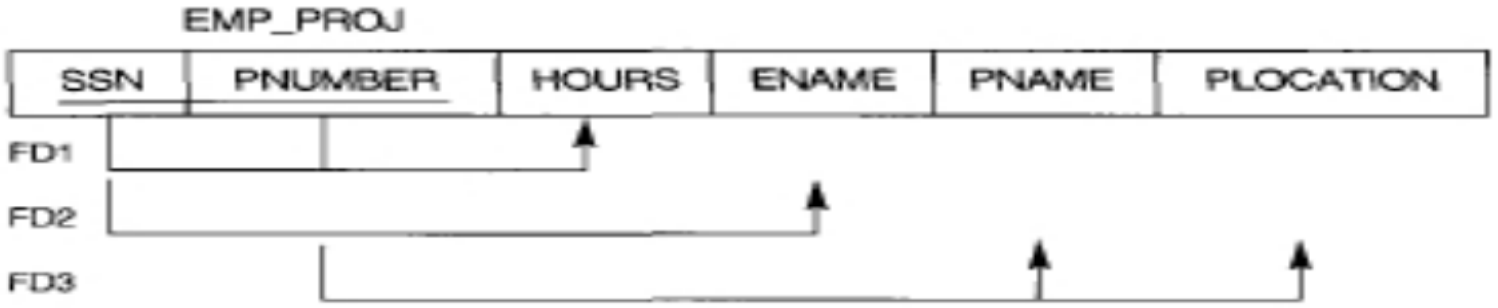
- To normalize this into INF, we remove the nested relation attributes into a new relation and propagate the primary key into it; the primary key of the new relation will combine the partial key with the primary key of the original relation. Decomposition and primary key propagation yield the schemas

  EMP_PROJl and EMP_PROJ2.



Second Normal Form:

- Second normal form (2NF) is 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 does not hold any more. That is, for any attribute A € X, (X - {A}) does not functionally determine ‘Y’.
- A functional dependency X→Y is a partial dependency if some attribute A € X can be removed from ‘X’ and the dependency still holds. That is, for some A € X, (X - {A}) → Y.
- In the following {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.



- **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’. or 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’.**
- The test for 2NF involves testing for functional dependencies whose left-hand side is a primary key composed of multiple attributes. If the primary key contains a single attribute, the test need not be applied at all.
- The EMP_PROJ relation in the above figure is in 1NF but is not in 2NF.

a) The nonprime attribute ENAME violates 2NF because of FD2. ENAME is partially dependent on {SSN,PNUMBER} and not dependent on PNUMBER.(Given ENAME can be determined only by SSN. So the other attributes are not needed for that table)

b) The nonprime attributes PNAME and PLOCATION violates 2NF because of FD3. PNAME and PLOCATION are partially dependent on {SSN,PNUMBER} and not dependent on SSN.

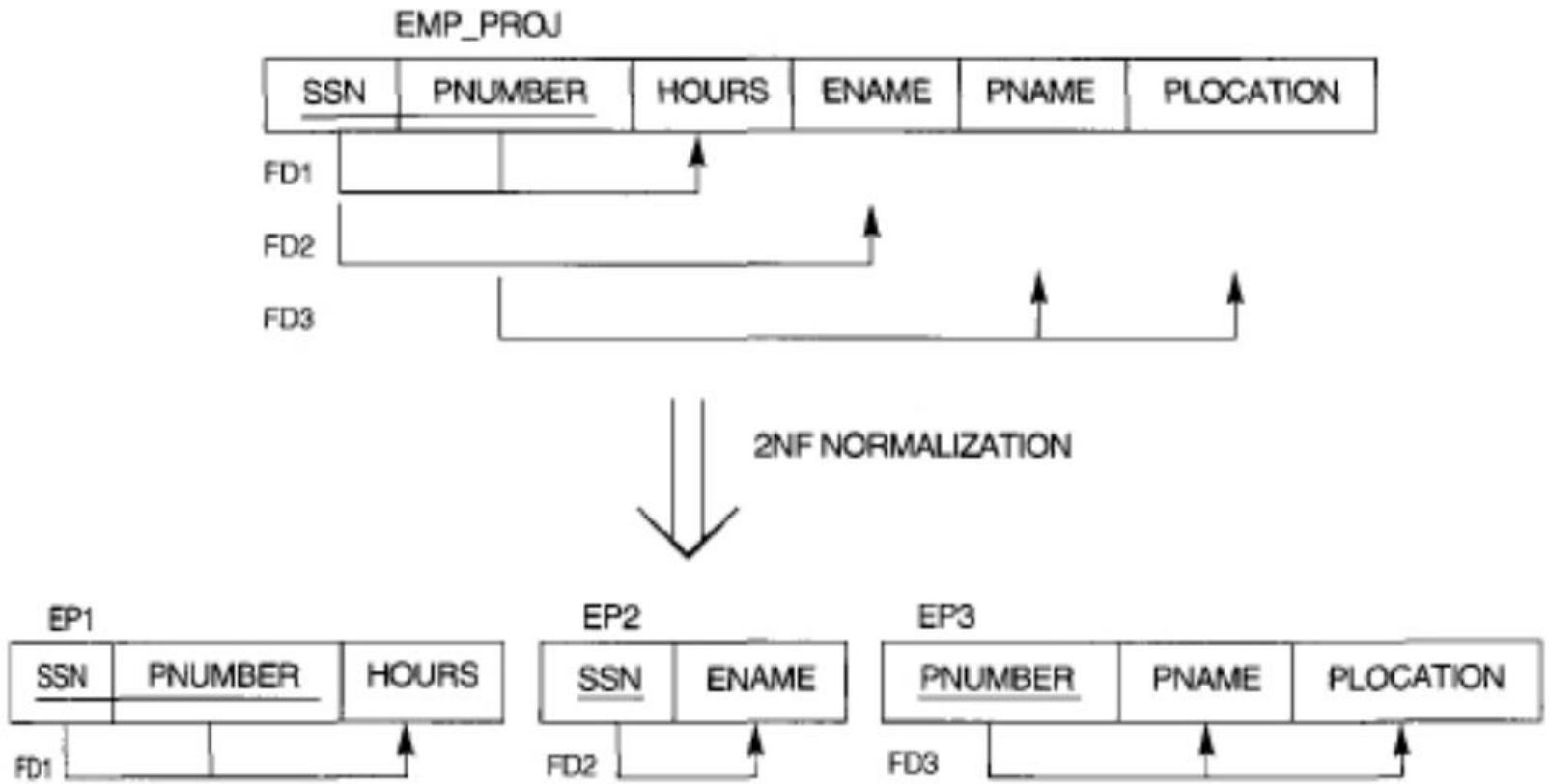
- The functional dependencies FD1, FD2 and FD3 in Figure ,hence lead to the decomposition of EMP_PROJ into the three relation schemas EPl, EP2, and EP3 shown below, each of which is in 2NF.

- The test for 2NF involves testing for functional dependencies whose left-hand side is a primary key composed of multiple attributes. If the primary key contains a single attribute, the test need not be applied at all.
- The EMP_PROJ relation in the above figure is in 1NF but is not in 2NF.

a) The nonprime attribute ENAME violates 2NF because of FD2. ENAME is partially dependent on {SSN,PNUMBER} and not dependent on PNUMBER.(Given ENAME can be determined only by SSN. So the other attributes are not needed for that table)

b) The nonprime attributes PNAME and PLOCATION violates 2NF because of FD3. PNAME and PLOCATION are partially dependent on {SSN,PNUMBER} and not dependent on SSN.

- The functional dependencies FD1, FD2 and FD3 in Figure 4.10 hence lead to the decomposition of EMP_PROJ into the three relation schemas EPl, EP2, and EP3 shown below, each of which is in 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:

- 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 is a set of attributes ‘Z’ that is neither a candidate key nor a subset of any key of R, and both X→Z and Z→Y hold.
- Definition: A relation schema ‘R’ is in 3NF if it satisfies 2NFand no nonprime attribute of ‘R’ is transitively dependent on the primary key. A relation schema ‘R’ is in third normal form (3NF) if, whenever a nontrivial functional dependency

  X → A holds in ‘R’, either

  (a) ‘X’ is a superkey of ‘R’, or

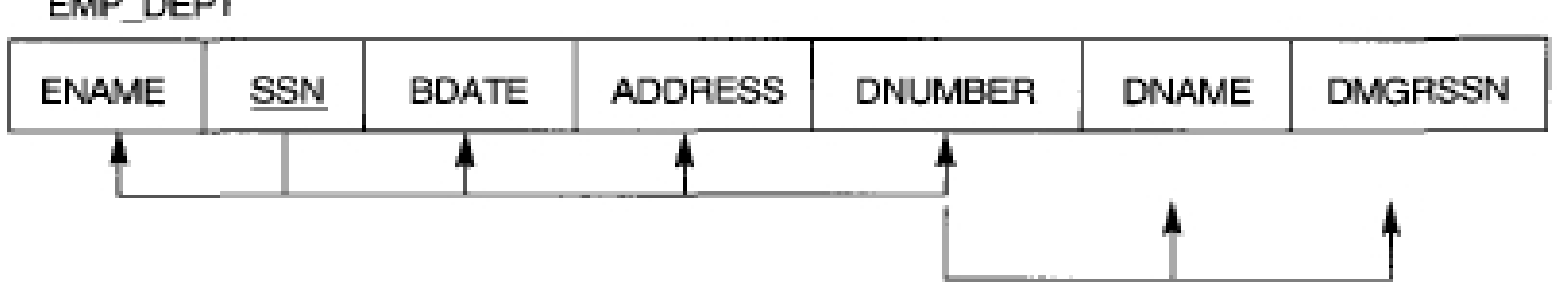
  (b) ‘A’ is a prime attribute of R

- The dependency SSN→ DMGRSSN is transitive through DNUMBER in EMP_DEPT of Figure because:

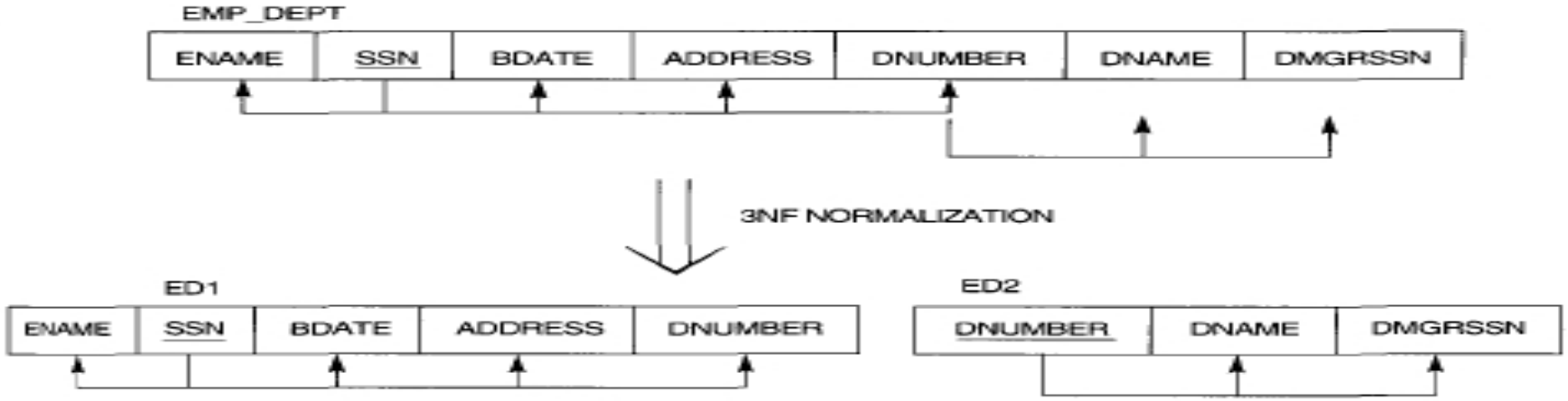
  a) Both the dependencies SSN→DNUMBER and DNUMBER→DMGRSSN hold.

  b) DNUMBER is neither a key itself nor a subset of the key of EMP_DEPT.

  c) We can see that the dependency of Dmgr_ssn on Dnumber is undesirable in EMP_DEPT since Dnumber is not a key of EMP_DEPT.



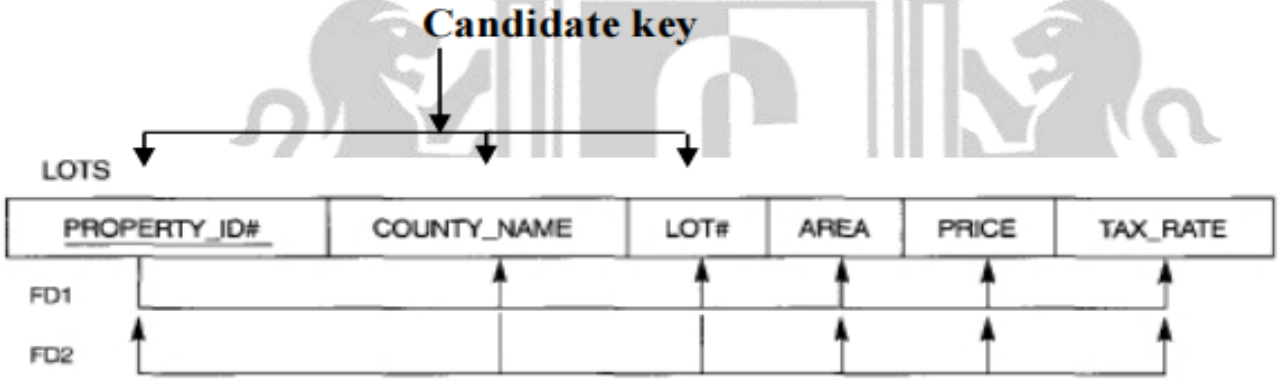
The relation schema EMP_DEPT in Figure is in 2NF, since no partial dependencies on a key exist. However, EMP_DEPT is not in 3NF because of the transitive dependency of DMGRSSN (and also DNAME) on SSN via DNUMBER. We can normalize EMP_DEPT by decomposing it into the two 3NF relation schemas EDl and ED2 shown in following Figure



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).

EXAMPLE:

- Suppose that there are two candidate keys: 1) PROPERTY_ID# and 2) {COUNTY_NAME, LOT#}; that is, lot numbers are unique only within each county, but PROPERTY ID numbers are unique across counties for the entire state.

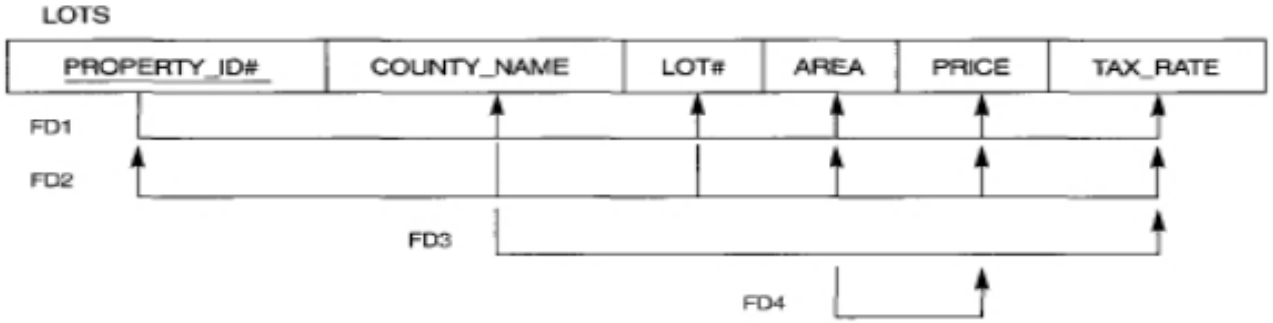


Based on the two candidate keys PROPERTY_ID# and {COUNTY_NAME, LOT#}, the functional dependencies FD1 and FD2 of Figure hold.

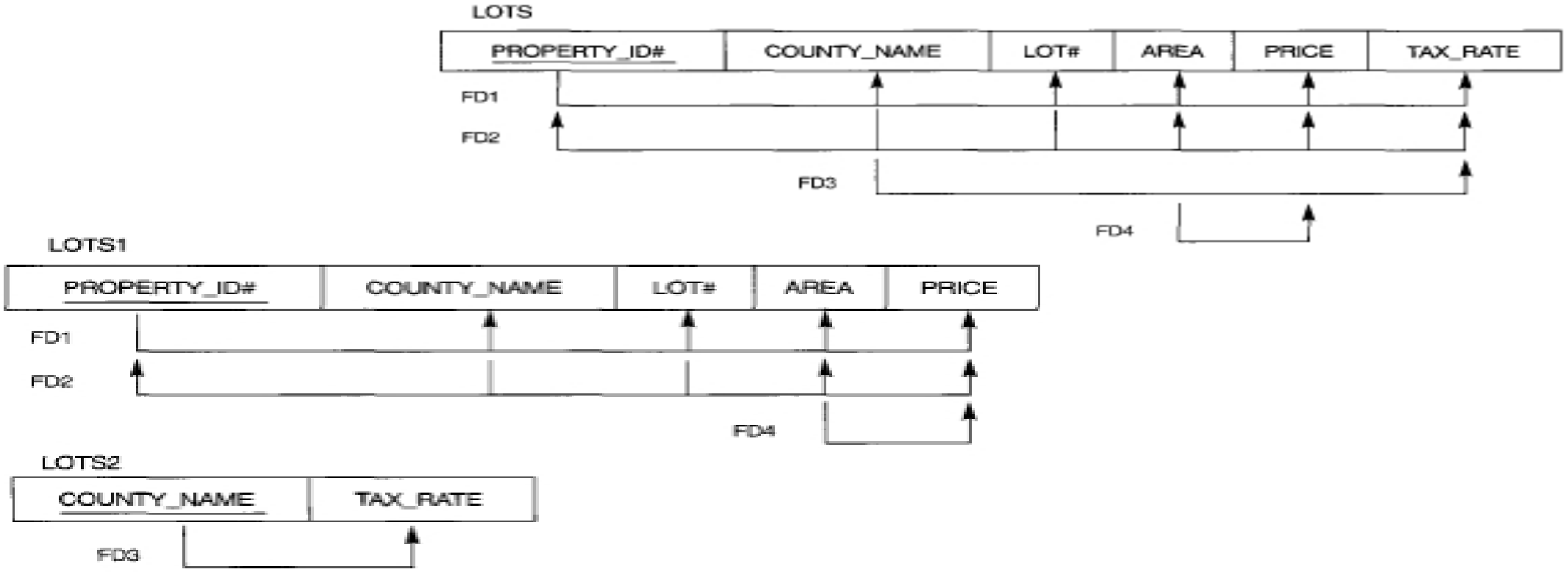
We choose PROPERTY_ID# as the primary key, so it is underlined in Figure . Suppose that the following two additional functional dependencies hold in LOTS:

FD3: COUNTY_NAME → TAX_RATE

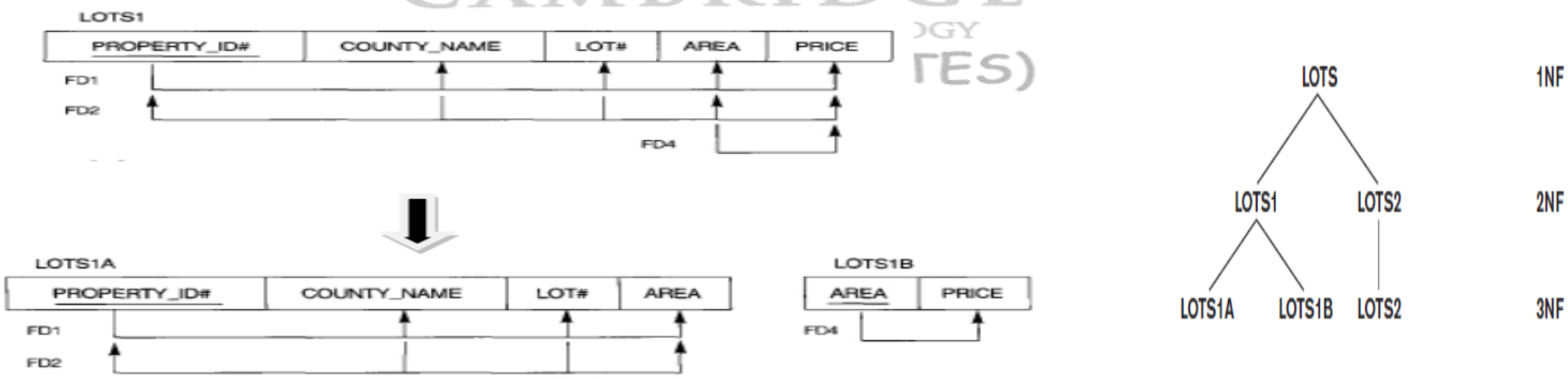
FD4: AREA → PRICE



- The LOTS relation schema violates the general definition of 2NF as TAX_RATE is partially dependent on the candidate key {COUNTY_NAME, LOT#} because:
    a) Due to FD2 {COUNTY_NAME, LOT#} → TAX_RATE
    b) Due to FD3 {COUNTY_NAME} → TAX_RATE
- To normalize LOTS into 2NF, we decompose it into the two relations LOTS1 and LOTS2, shown below.



- To normalize LOTS1 into 3NF, we decompose it into the relation schemas LOTS1A and LOTS1B as shown in Figure



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

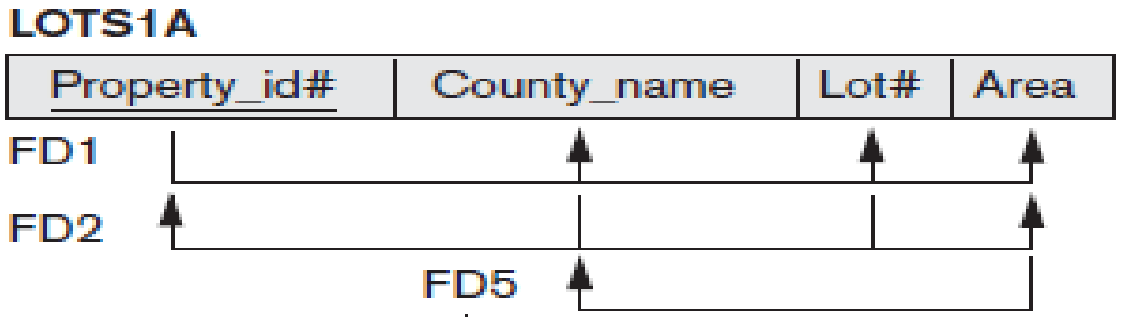
 Two points are worth noting about this example and the general definition of 3NF:

a) LOTS1 violates 3NF because PRICE is transitively dependent on each of the candidate keys of LOTS1 via the nonprime attribute AREA.

b) we find that both FD3 and FD4 violate 3NF. We could hence decompose LOTS into LOTSlA, LOTSlB, and LOTS2 directly

BOYCE-CODD NORMAL FORM:

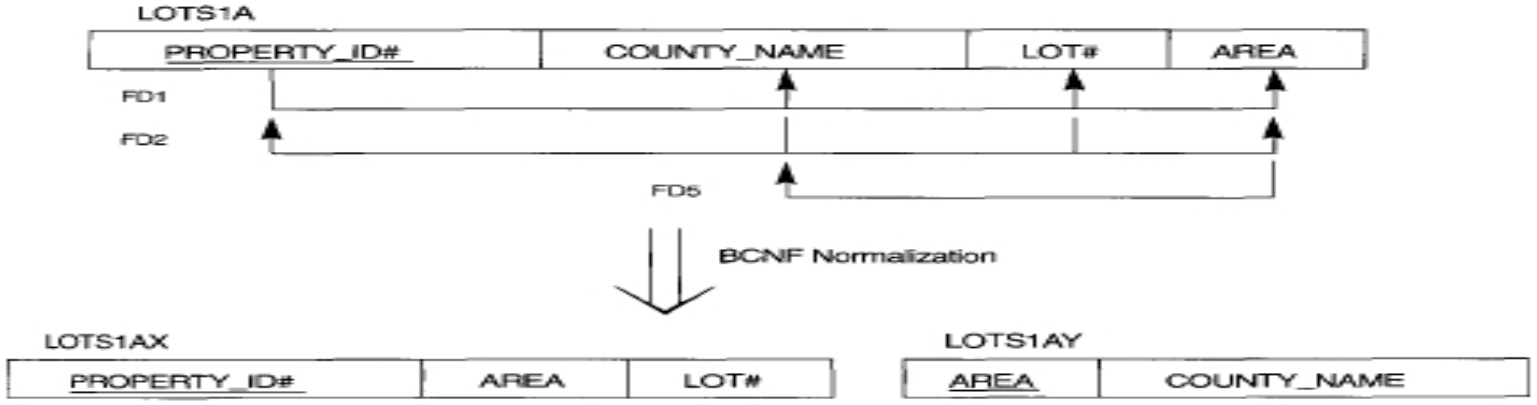
- Boyce-Codd normal form (BCNF) was proposed as a simpler form of 3NF, but it was found to be stricter than 3NF.
- Every relation in BCNF is also in 3NF; however, a relation in 3NF is not necessarily in BCNF.
- **Definition. A relation schema R is in BCNF if whenever a nontrivial functional dependency X → A holds in R, then X is a superkey of R.**



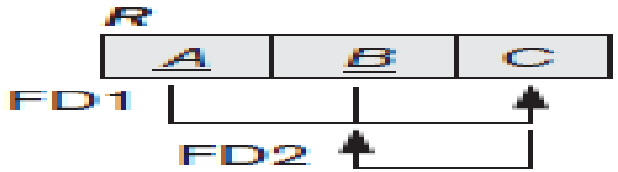
FD5: AREA → COUNTY_NAME

 The relation schema LOTS1A still is in 3NF because COUNTY_NAME is a prime attribute.

- The only difference between the definitions of BCNF and 3NF is that condition (b) of 3NF, which allows A to be prime, is absent from BCNF.
- In our example, FD5 violates BCNF in LOTS1A because AREA is not a superkey of LOTS1A.
- Note that FD5 satisfies 3NF in LOTSIA because COUNTY_NAME is a prime attribute (condition b), but this condition does not exist in the definition of BCNF.
- We can decompose LOTSIA into two BCNF relations LOTS1AX and LOTS1AY as shown below.



The relation schema R shown in following Figure illustrates the general case of a relation being in 3NF but not in BCNF.



MULTIVALUED DEPENDENCY (MVD) 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.

Formal Definition of Multivalued Dependency

- Definition of Multivalued dependency: A multivalued dependency 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 t1 and t2 exist in r such that t1[X] = t2[X], then two tuples t3 and t4 should also exist in r with the following properties, where we use Z to denote (R – (X ∪ Y)):

  t3[X] = t4[X] = t1[X] = t2[X].

  t3[Y] = t1[Y] and t4[Y] = t2[Y].

  t3[Z] = t2[Z] and t4[Z] = t1[Z].

- Whenever x->->Y holds, we say that X multidetermines Y. Because of the symmetry in the definition, whenever x->->Y holds in R, so does x->->Y. Hence, x->->Yimplies x->->Y, and therefore it is sometimes written as x->->Y.

- Definition of 4NF: 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 →→ Y in F+ X is a superkey for R.

Inference rules followed in 4NF are:

IR1 (reflexive rule for FDs): If $X \supseteq Y$, then $X \rightarrow Y$.
IR2 (augmentation rule for FDs): $\{X \rightarrow Y\} \models XZ \rightarrow YZ$.
IR3 (transitive rule for FDs): $\{X \rightarrow Y, Y \rightarrow Z\} \models X \rightarrow Z$.
IR4 (complementation rule for MVDs): $\{X \twoheadrightarrow Y\} \models \{X \twoheadrightarrow (R - (X \cup Y))\}$.
IR5 (augmentation rule for MVDs): If $X \twoheadrightarrow Y$ and $W \supseteq Z$, then $WX \twoheadrightarrow YZ$.
IR6 (transitive rule for MVDs): $\{X \twoheadrightarrow Y, Y \twoheadrightarrow Z\} \models X \twoheadrightarrow (Z - Y)$.
IR7 (replication rule for FD to MVD): $\{X \rightarrow Y\} \models X \twoheadrightarrow Y$.
IR8 (coalescence rule for FDs and MVDs): If $X \twoheadrightarrow Y$ and there exists W with the properties that (a) $W \cap Y$ is empty, (b) $W \rightarrow Z$, and (c) $Y \supseteq Z$, then $X \rightarrow Z$.

- In the EMP relation of Figure 4.15(a), the values ‘X’ and ‘Y’ of Pname are repeated with each value of Dname (or, by symmetry, the values ‘John’ and ‘Anna’ of Dname are repeated with each value of Pname). In 4.15 (c), not every Sname determines various Part_name and not every Sname determines multiple Proj_name. so it is not MVD. Therefore it is in 4NF.

Example 1: Figure 4.15

Fourth and fifth normal forms.
(a) The EMP relation with two MVDs: Ename →→ Pname and Ename →→ Dname.
(b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

(a) **EMP**

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

(b) **EMP_PROJECTS**

Ename	Pname
Smith	X
Smith	Y

EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

(c) **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

Decomposing a relation state of EMP that is not in 4NF. (a) EMP relation with additional tuples. (b) Two corresponding 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.

(a) **EMP**

Ename	Pname	Dname
Smith	X	John
Smith	Y	Anna
Smith	X	Anna
Smith	Y	John
Brown	W	Jim
Brown	X	Jim
Brown	Y	Jim
Brown	Z	Jim
Brown	W	Joan
Brown	X	Joan
Brown	Y	Joan
Brown	Z	Joan
Brown	W	Bob
Brown	X	Bob
Brown	Y	Bob
Brown	Z	Bob

(b) **EMP_PROJECTS**

Ename	Pname
Smith	X
Smith	Y
Brown	W
Brown	X
Brown	Y
Brown	Z

EMP_DEPENDENTS

Ename	Dname
Smith	Anna
Smith	John
Brown	Jim
Brown	Joan
Brown	Bob

JOIN DEPENDENCIES AND FIFTH NORMAL FORM (5NF)

- Definition of join dependency: A join dependency (JD), denoted by JD(R1, R2, ..., Rn), 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 R1, R2, ..., Rn.
- Hence, for every such r we haveDefinition of 5 NF: 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(R1, R2, ..., Rn) in F+ (that is, implied by F), every Ri is a superkey of R.
- Figure 4.16(d) shows how the SUPPLY relation with the join dependency is decomposed into three relations R1, R2, and R3 that are each in 5NF.
- 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.

(c) The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R_1, R_2, R_3).
(d) Decomposing the relation SUPPLY into the 5NF relations R_1, R_2, R_3.

(c) **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

(d) 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

Input: A relation R and a set of functional dependencies F on the attributes of R.
1. Set K := R.
2. For each attribute A in K
{compute (K – A)+ with respect to F;
if (K – A)+ contains all the attributes in R, then set K := K – {A} };