Relational Design Algorithms

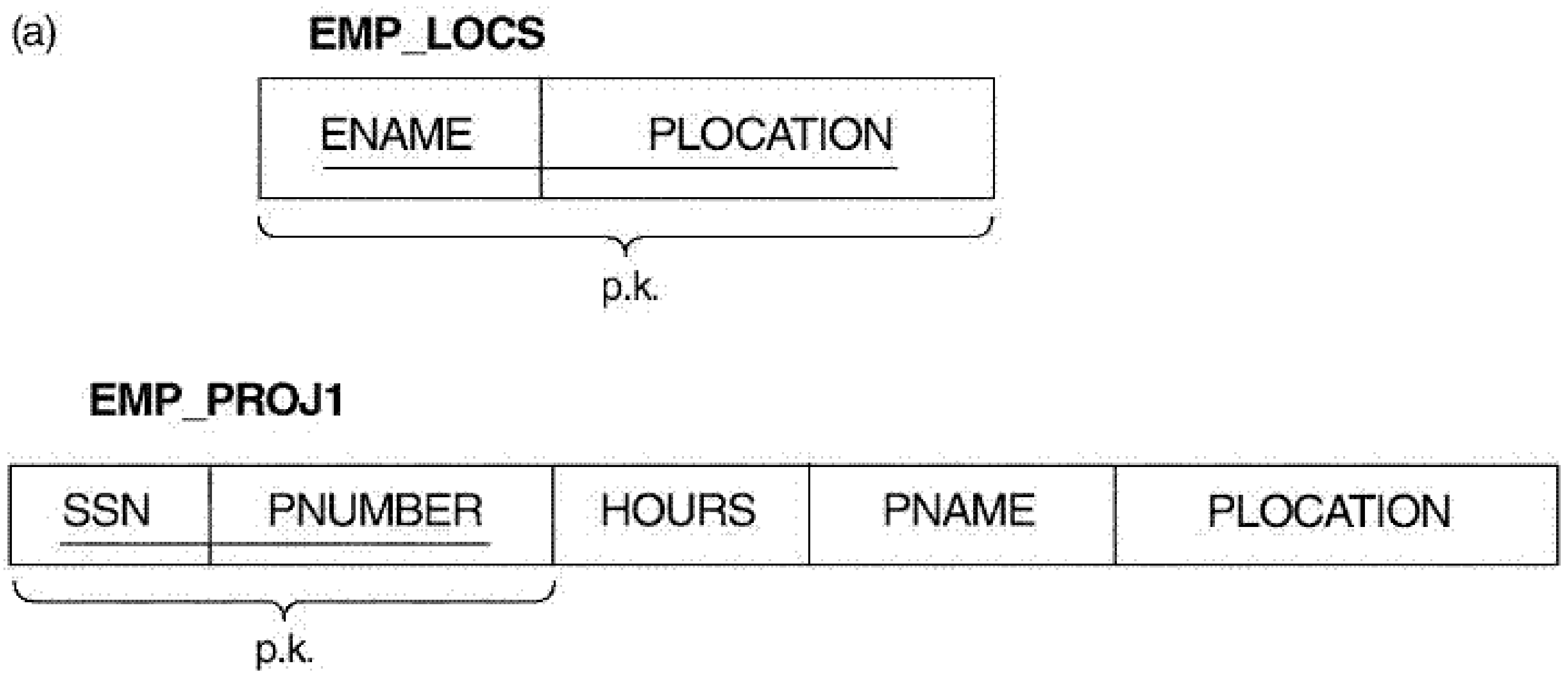
Why Design Algorithms?

- An individual relation in a higher normal form does not, on its own, guarantee a good design.
- A set of relations that together form the relational database schema must possess certain additional properties to ensure a good design. We discuss two of them:

1. **The dependency preservation property**
2. **The lossless or nonadditive join property.**

Why Design Algorithms?

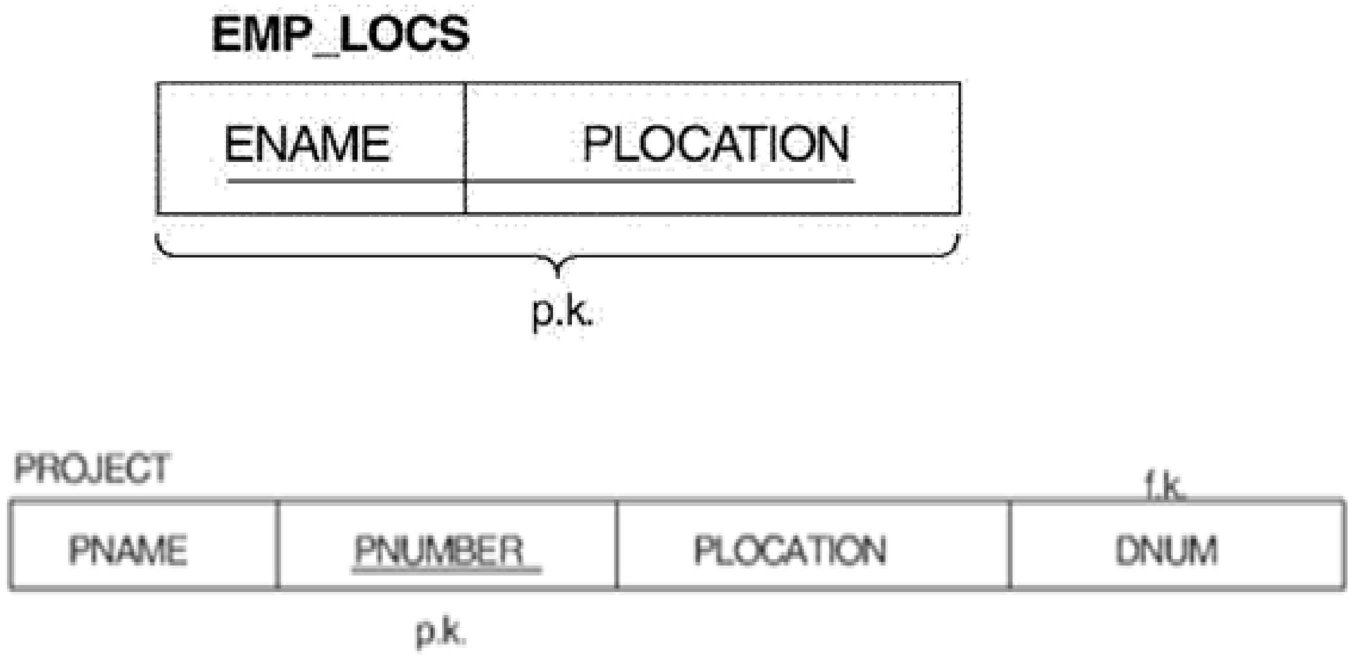
Example :



Although EMP_LOCS is in BCNF, it still gives rise to spurious tuples when joined with EMP_PROJ1

Why Design Algorithms?

Example :



Joining EMP_LOCS with PROJECT which is in BCNF-also gives rise to spurious tuples

PROPERTIES OF RELATIONAL DECOMPOSITIONS

Assumptions

1. A **single universal relation schema** R = {A1, A2, ... An} that includes all the attributes of the database.
2. Every attribute name is unique.
3. The set F of functional dependencies that should hold on the attributes of R is specified by the database designers and is made available to the design algorithms.

Attribute preservation condition of a decomposition

- The algorithms decompose the universal relation schema R into a set of relation schemas D = {R1, R2 ... , Rm} that will become the relational database schema; **D is called a decomposition of R**.

- Each attribute in R will appear in at least one relation schema R_i in the decomposition so that no attributes are "lost";

$$\boxed{\bigcup_{i=1}^{m} R_i = R}$$

1. Dependency Preservation Property of a Decomposition

What is dependency preservation condition?

- When we decompose R into a set of relation schemas

  D = {R_1,R_2…R_n} , we want each functional dependency

  $X \rightarrow A$ in F to be in one of the relation schemas **R_i**.

This is the **dependency preservation condition.**

Why we want to preserve the dependencies ?

- Each dependency in F represents a constraint on the database.
- If one of the dependencies is not present in D then

1. We have to join two or more of the relations in the decomposition
2. And then check that the functional dependency holds in the result of the JOIN operation.

Note the following:

1. It is not necessary that the exact dependencies specified in F appear themselves in individual relations of the decomposition D.
2. **It is sufficient that the union of the dependencies that hold on the individual relations in D be equivalent to F.**

Formal Definition

Let of F be a set of dependencies of on R.

Let π_{R_i} **(F)** denote the **projection** of F on R_i , where R_i is a subset of R

(Projection of F on R_i is the set of dependencies X →Y in F^+ such that the attributes in X U Y are all contained in R_i)

Then,

A decomposition D = {R1, R2, ... , Rm} of R is **dependency-preserving** with respect to F if the union of the projections of F on each R_i in D is equivalent to F;

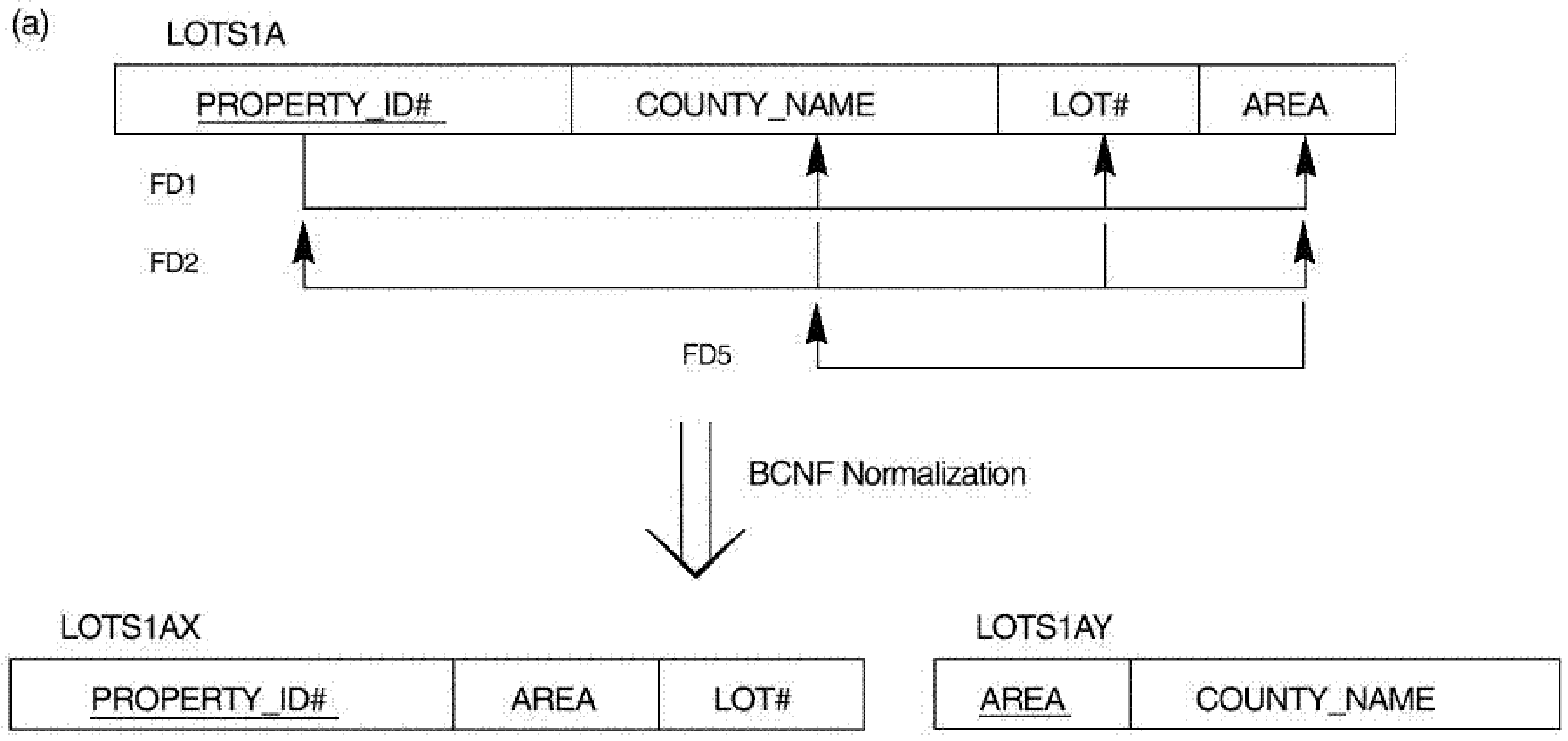
$$((\pi_{R_1}(F)) \cup \ldots \cup (\pi_{R_m}(F)))^+ = F^+$$

Or $(F1 \cup F2 \cup F3 \ldots \cup Fm)^+ = F^+$

where F1 is FDs of R1 and F2 is FDs of R2 and so

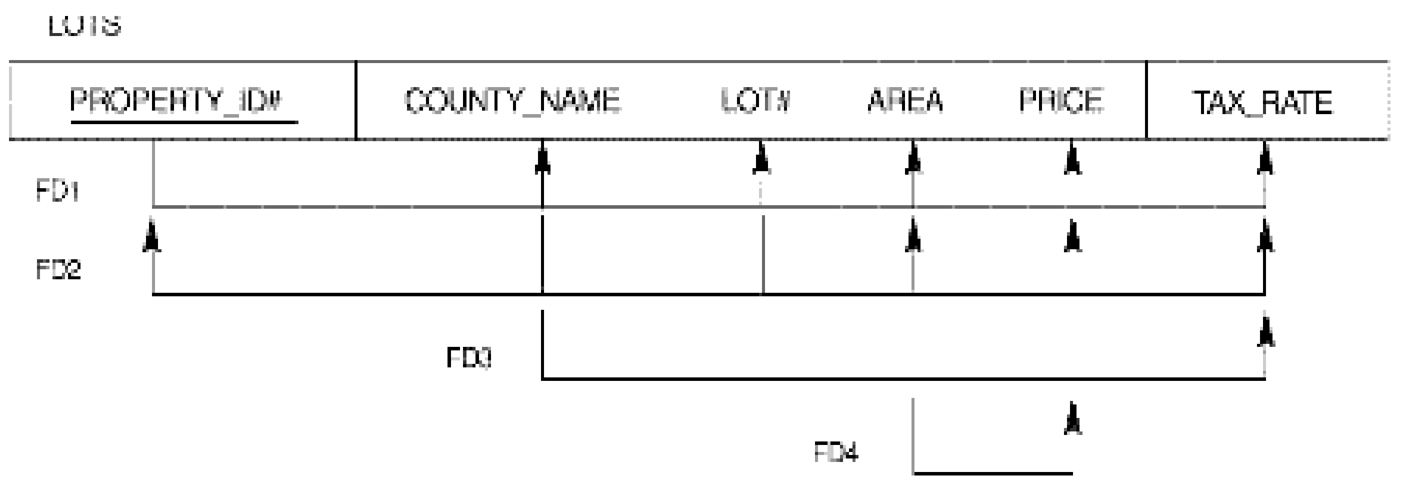
Example 1

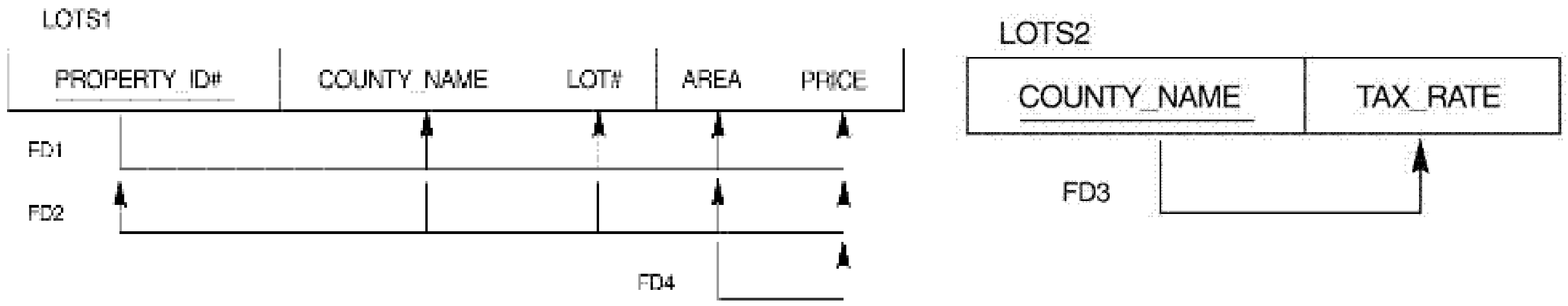
The following decomposition does not preserve dependencies.(FD2 is lost)



Example 2

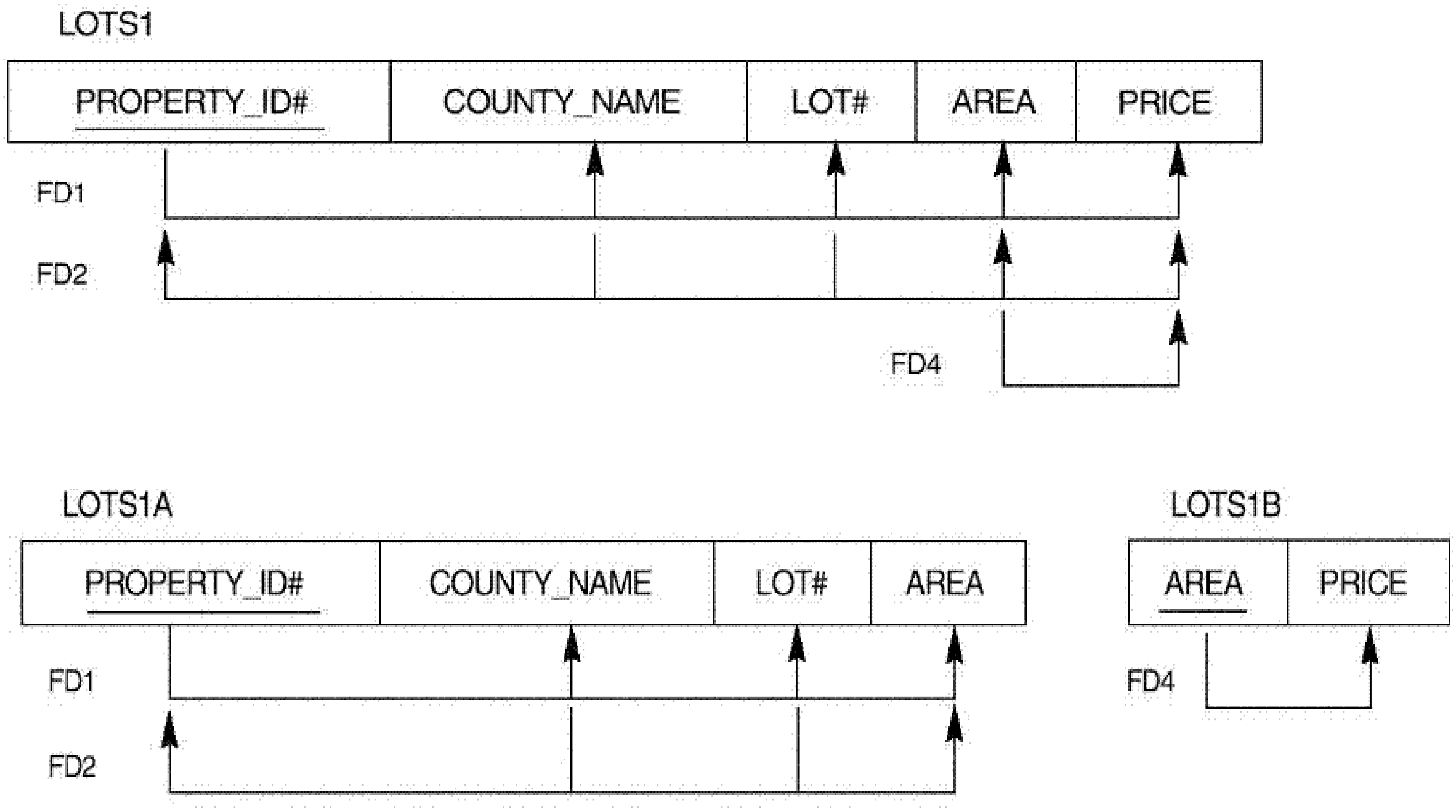
- The following decompositions preserve all the dependencies.





Example 3

- The following decompositions preserve all the dependencies.



Example 4

All three decompositions "lose" the functional dependency FD1.

Consider,

FD1: {STUDENT, COURSE} → INSTRUCTOR

FD2: INSTRUCTOR → COURSE

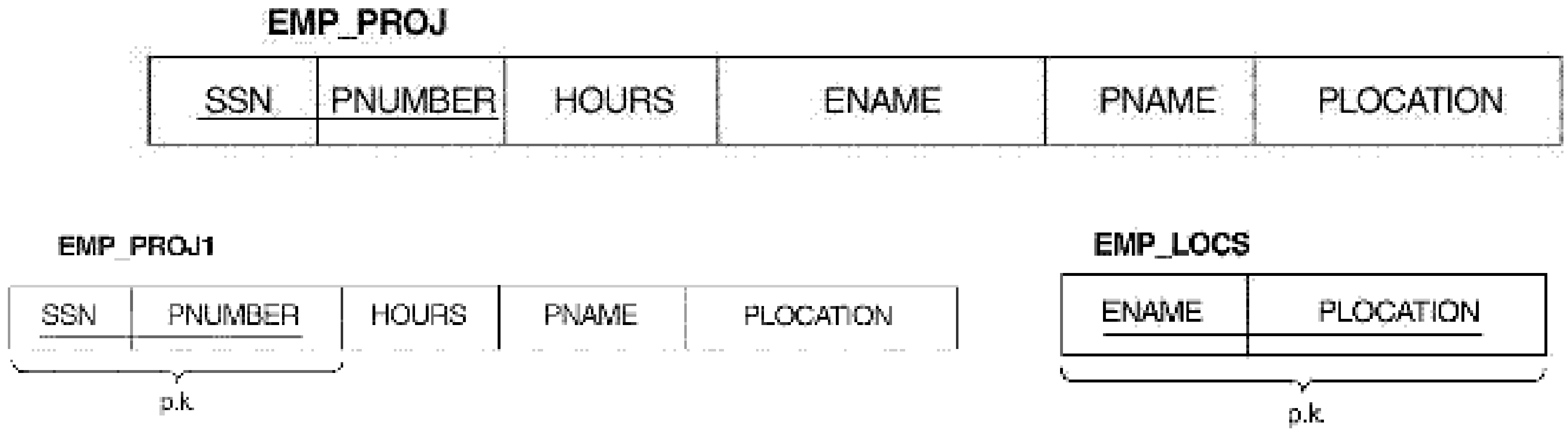
3 possible decompositions of the above relation is,

1. {INSTRUCTOR , COURSE} and {INSTRUCTOR, STUDENT}
2. {STUDENT, INSTRUCTOR} and {STUDENT , COURSE}.
3. {COURSE , INSTRUCTOR} and {COURSE , STUDENT}.

2. Lossless (Nonadditive) Join Property of a Decomposition

- The lossless join or nonadditive join property, ensures that no spurious tuples are generated when a NATURAL JOIN operation is applied to the relations in the decomposition.

Example : The following decomposition generates spurious tuples when we apply natural join (*)



2. Lossless (Nonadditive) Join Property of a Decomposition – Example for decomposition which is not lossless

R

Model Name	Price	Category
a11	100	Canon
s20	200	Nikon
a70	150	Canon

$\downarrow$

R1

Model Name	Category
a11	Canon
s20	Nikon
a70	Canon

R2

Price	Category
100	Canon
200	Nikon
150	Canon

2. Lossless (Nonadditive) Join Property of a Decomposition – Example for decomposition which is not lossless

R1 * R2

Model Name	Price	Category
a11	100	Canon
a11	150	Canon
s20	200	Nikon
a70	100	Canon
a70	150	Canon

If the two decomposed relation were related by Model Name , the we would have obtained :

R1 * R2

Model Name	Price	Category
a11	100	Canon
s20	200	Nikon
a70	150	Canon

2. Lossless (Nonadditive) Join Property of a Decomposition

Formal Definition

Formally, a decomposition D = {R1, R2, . . . , Rm) of R has the **lossless (nonadditive) join property** with respect to the set of dependencies F on R if, for every relation state r of R that satisfies F, the following holds, where * is the NATURAL JOIN of all the relations in D:

$$\boxed{*\,(\pi_{R_1}(r), \ldots, \pi_{R_m}(r)) = r}$$

OR $(\; r_1 * r_2 * r_3 \ldots \; * r_m) \; = r$

where r_1 is relation state of R_1 , r_2 is relation state of R_2 and so on

The lossless join property is always defined with respect to a specific set F of dependencies.

Testing for Lossless Join Property of a Decomposition

Algorithm : Testing for Lossless (nonadditive) Join Property

Input: A universal relation R, a decomposition D = $\{R_1, R_2, \ldots R_m\}$ of R, and a set F of functional dependencies.

1. Create an initial matrix S with one row *i for each relation* R_i *in D, and one column* j for each attribute A_j in R.
2. Set $S(i, j) := b_{i,j}$ *for all matrix entries.*

   (* each $b_{i,j}$ is a distinct symbol associated with indices *(i,j)* *)

3. For each row i representing relation schema R_i

   {for each column j representing attribute A_j

   {if (relation R_i, includes attribute A_j) then set $S(i, j) := a_j$;};};

   (* each a_j is a distinct symbol associated with index (j) *)

Testing for Lossless Join Property of a Decomposition

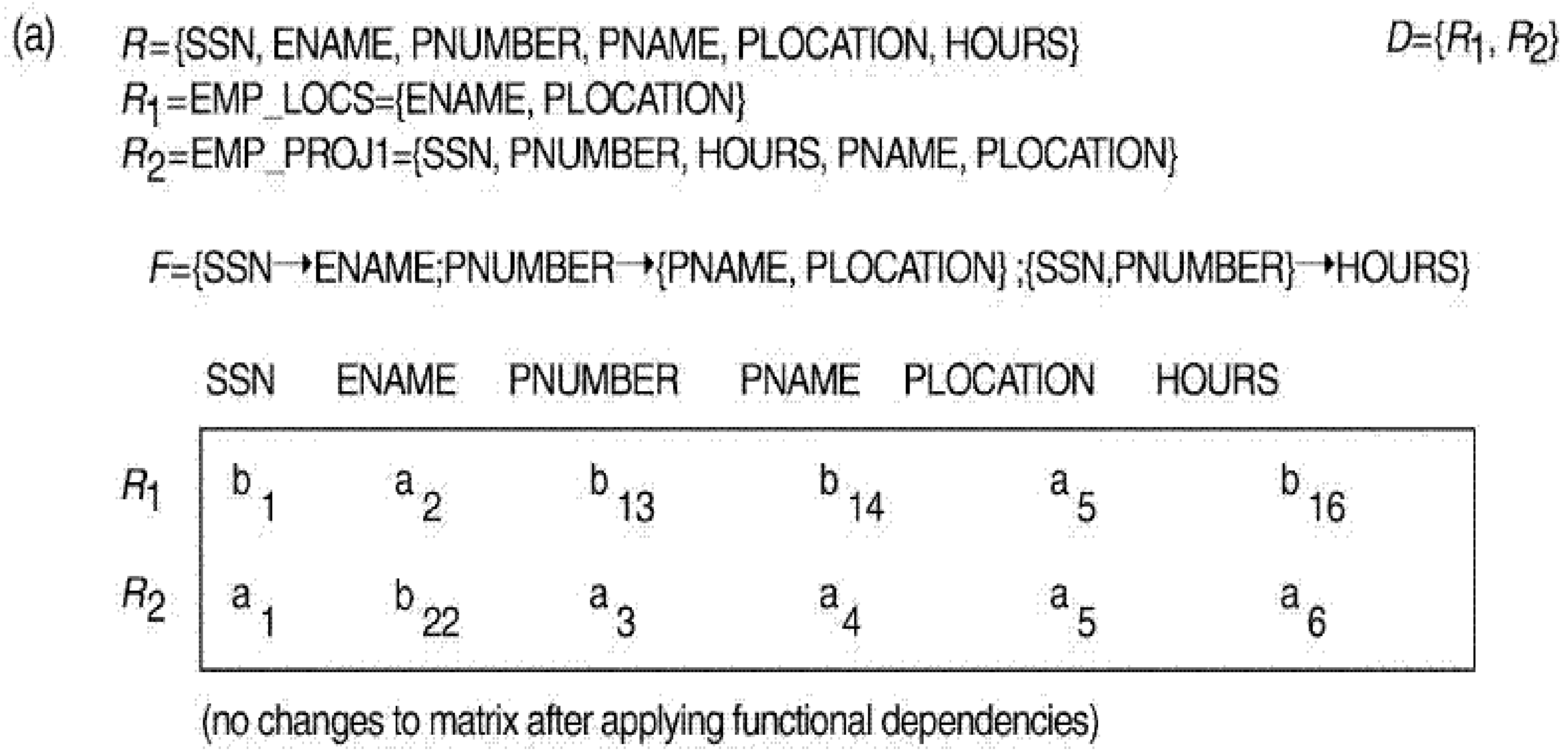
4. Repeat the following loop until a *complete loop execution results in no changes to S*

```
    {for each functional dependency X → Y in F
        {for all rows in S that have the same symbols in the columns corresponding to attributes in X
            {make the symbols in each column that correspond to an attribute in Y be the same in all these rows as follows:
                If any of the rows has an "a" symbol for column, set the other rows to that same "a" symbol in the column. If no "a" symbol exists for the attribute in any of the rows, choose one of the "b" symbols that appears in one of the rows for the attribute and set the other rows to that same "b" symbol in the column ;};};};
```

5. If a row is made up entirely of *"a" symbols, then the decomposition has the lossless* join property; otherwise, it does not.

Testing for Lossless Join Property of a Decomposition

Example 1 :



(a) $R=\{SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS\}$ $D=\{R_1, R_2\}$

R_1=EMP_LOCS={ENAME, PLOCATION}

R_2=EMP_PROJ1={SSN, PNUMBER, HOURS, PNAME, PLOCATION}

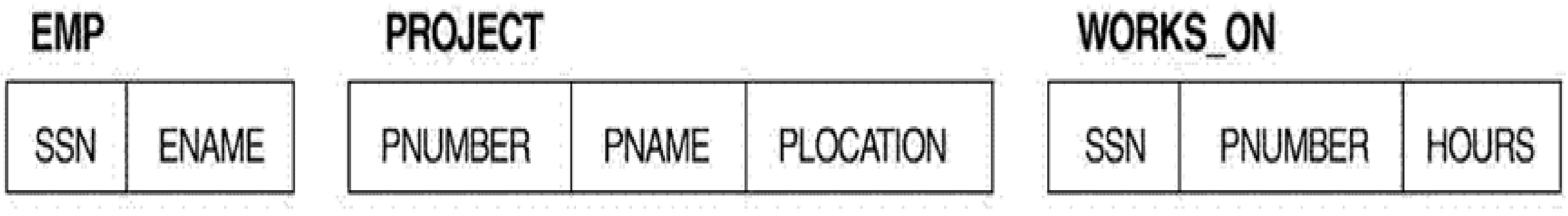
$F=\{SSN \rightarrow ENAME; PNUMBER \rightarrow \{PNAME, PLOCATION\}; \{SSN, PNUMBER\} \rightarrow HOURS\}$

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	b_1	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

(no changes to matrix after applying functional dependencies)

Testing for Lossless Join Property of a Decomposition

Example 2 :



EMP

SSN	ENAME

PROJECT

PNUMBER	PNAME	PLOCATION

WORKS_ON

SSN	PNUMBER	HOURS

(c) R={SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} $D=\{R_1, R_2, R_3\}$

R_1=EMP={SSN, ENAME}

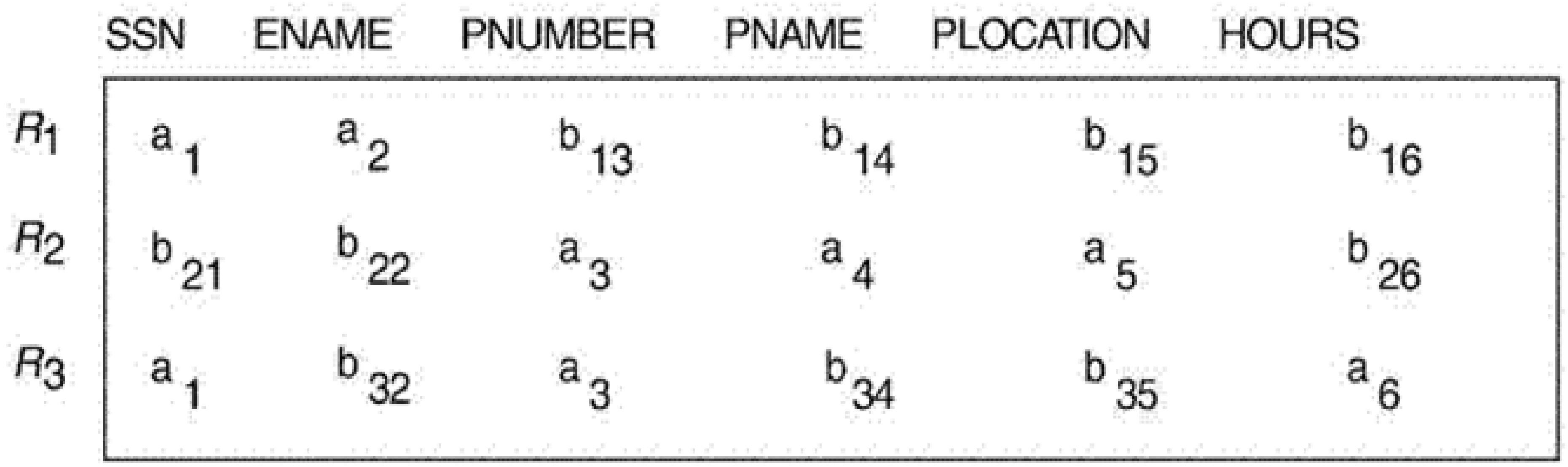
R_2=PROJ={PNUMBER, PNAME, PLOCATION}

R_3=WORKS_ON={SSN, PNUMBER, HOURS}

F={SSN $\rightarrow$ ENAME;PNUMBER $\rightarrow$ {PNAME, PLOCATION} ;{SSN,PNUMBER} $\rightarrow$ HOURS}

Testing for Lossless Join Property of a Decomposition

Example 2:



	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	b_{32}	a_3	b_{34}	b_{35}	a_6

(original matrix S at start of algorithm)

Testing for Lossless Join Property of a Decomposition

Example 2:

	SSN	ENAME	PNUMBER	PNAME	PLOCATION	HOURS
R_1	a_1	a_2	b_{13}	b_{14}	b_{15}	b_{16}
R_2	b_{21}	b_{22}	a_3	a_4	a_5	b_{26}
R_3	a_1	$\cancel{b_{32}}$ a_2	a_3	$\cancel{b_{34}}$ a_4	$\cancel{b_{35}}$ a_5	a_6

(matrix S after applying the first two functional dependencies - last row is all "a" symbols, so we stop)

The lossless join testing algorithm. (a) Applying the algorithm to test the decomposition of EMP_PROJ into EMP_PROJ1 and EMP_LOCS. (b) Another decomposition of EMP_PROJ. (c) Applying the algorithm to the decomposition in Figure 15.01(b).

Testing for Lossless Join Property of a Decomposition

Exercise

Consider the Relation Schema R = (A, B, C, D, E, G) and the FD set

$F = \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G\}$.

Determine whether the following decomposition of R has the lossless join property with respect to F.

R1 = {A, B, C} R2 = {A, C, D, E} R3 = {A, D, G}

Testing Binary Decompositions for the lossless Join Property:

For Binary decomposition, the following test is sufficient to ensure that the decomposition has lossless join property.

PROPERTY LJ1 (LOSSLESS JOIN TEST FOR BINARY DECOMPOSITIONS)

A decomposition D = {R1,R2} of R has the lossless (nonadditive) join property with respect to a set of functional dependencies F on R ***if and only if either***

- **The FD ((R 1 I∩) → (R1 – R2)) is in F^+, OR**

- **The FD ((R1 I∩) → (R2 – R1)) is in F^+,**

Testing Binary Decompositions for the lossless Join Property:

Exercise

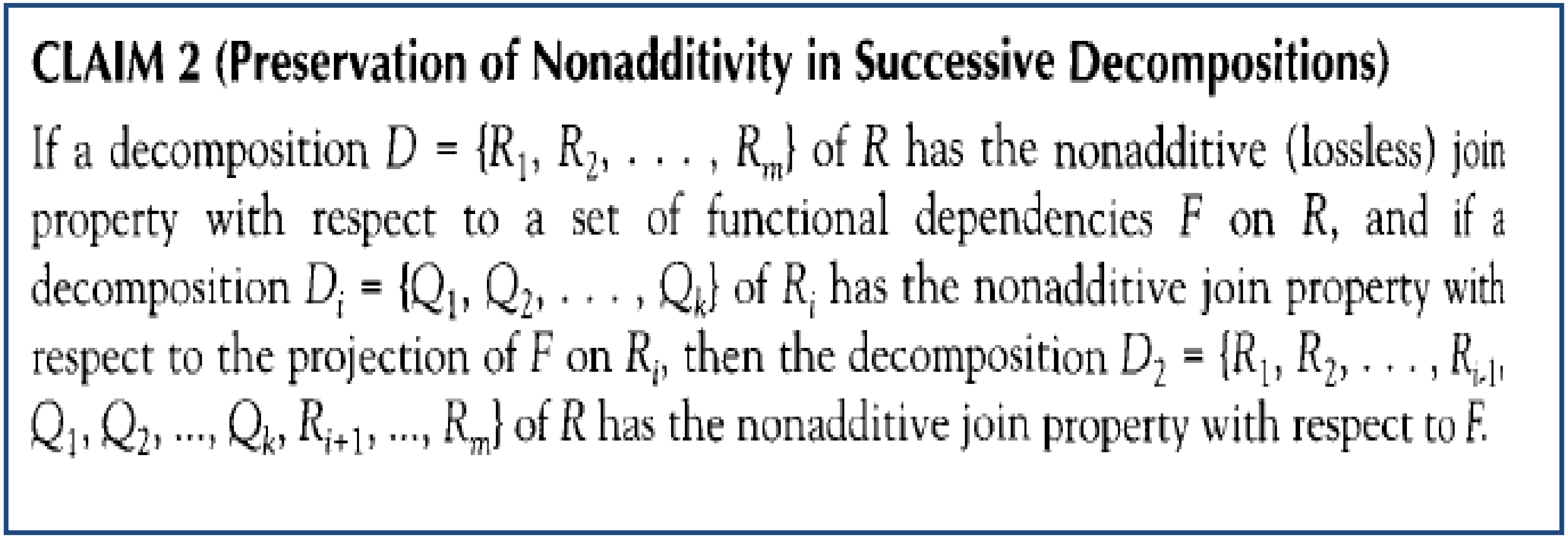
Consider the Relation Schema R = (A, B, C, D, E, G) and the FD set

$F = \{AB \rightarrow C, AC \rightarrow B, AD \rightarrow E, B \rightarrow D, BC \rightarrow A, E \rightarrow G\}$.

Determine whether the following binary decomposition of R has the lossless join property with respect to F.

1. **R1 = {A, B, C} and R2 = {A, C, D, E,G}**
2. **R1 = {A, B,C, D, E} and R2 = {A, D, G}**
3. **R1 = { B, C,G} and R2 = {A,B,C, D,E}**

Successive Lossless (Nonadditive) Join Decompositions



CLAIM 2 (Preservation of Nonadditivity in Successive Decompositions)

If a decomposition $D = \{R_1, R_2, \ldots, R_m\}$ of R has the nonadditive (lossless) join property with respect to a set of functional dependencies F on R, and if a decomposition $D_i = \{Q_1, Q_2, \ldots, Q_k\}$ of R_i has the nonadditive join property with respect to the projection of F on R_i, then the decomposition $D_2 = \{R_1, R_2, \ldots, R_{i-1}, Q_1, Q_2, \ldots, Q_k, R_{i+1}, \ldots, R_m\}$ of R has the nonadditive join property with respect to F.

Three algorithms for creating a relational decomposition.

1. **Relational Synthesis into 3NF with Dependency Preservation**
2. **Relational Decomposition into BCNF with Nonadditive Join Property**
3. **Relational Synthesis into 3NF with Dependency Preservation and Nonadditive (Lossless) Join Property**

1. Relational Synthesis into 3NF with Dependency Preservation

- The algorithm creates a dependency-preserving decomposition D = {R1, R2, ... , Rm} of a universal relation R based on a set of functional dependencies F, such that each R_i, in D is in 3NF.

- It guarantees only the dependency-preserving property; it does not guarantee the lossless join property.

Algorithm: Relational Synthesis into 3NF with Dependency Preservation

Input: A universal relation R and a set of functional dependencies F on the attributes of R.

1. Find a minimal cover G for F ;
2. For each left-hand-side X of a functional dependency that appears in G, create a relation schema in D with attributes $\{X \cup \{A_1\} \cup \{A_2\} \ldots \cup \{A_k\}\}$, *where* $X \rightarrow A_1, X \rightarrow A_2, \ldots X \rightarrow A_k$ *are the only dependencies in G with X as the left-hand-side .* (X is the key of this relation);
3. Place any remaining attributes in a single relation schema to ensure the attribute preservation property.

Exercise -1

Consider the universal relation R = (A, B, C, D, E, F, G, H, I, J) and the set of functional dependencies

$F = \{AB \rightarrow C, A \rightarrow DE, B \rightarrow F, F \rightarrow GH, D \rightarrow IJ\}$.

Decompose the above relation into 3NF using Relational Synthesis with Dependency- Preservation algorithm

Exercise - 2

Consider the universal relation R = {A, B, C, D, E, F, G, H, I, J}

and the set of functional dependencies

G = {AB→ C, BD→ EF, AD→GH, A→ I, H→J }.

Decompose the above relation into 3NF using Relational Synthesis with Dependency- Preservation algorithm

Relational Synthesis into 3NF with Dependency Preservation

Note the following:

1. It is obvious that all the dependencies in G are preserved by the algorithm because each dependency appears in one of the relations R_i in the decomposition D.
2. Since G is equivalent to F, all the dependencies in F are either preserved directly in the decomposition or are derivable using the inference rules.
3. The Algorithm is called the **relational synthesis algorithm**, because each relation schema R_i in the decomposition is synthesized (constructed) from the set of functional dependencies in G with the same left-hand-side X.

2. Relational Decomposition into BCNF with Nonadditive Join Property

- The algorithm decomposes a universal relation schema R = {A1,A2 … An} into a decomposition D={R1,R2, … ,Rm} such that each R_i is in BCNF and the decomposition D has the lossless join property with respect to F.

- The Algorithm utilizes Property LJ1 and Claim 2 (preservation of non additive in successive decompositions) to create a nonadditive join decomposition D

Algorithm: **Relational Decomposition into BCNF with Nonadditive Join Property**

Input: A universal relation R and a set of functional dependencies F on the attributes of R.

```
1. Set D = {R};
2. While there is a relation schema Q in D that is not in BCNF
   do {
        choose a relation schema Q in D that is not in BCNF;
        find a functional dependency X → Y in Q that violates
      BCNF
        replace Q in D by two relation schemas (Q - Y) and (X U Y)
;
      };
```

Exercise -1

Consider the universal relation R = {A, B, C, D, E, F, G, H, I, J}

and the set of functional dependencies

F = {AB→C, A→DE, B→F, F→GH, D→IJ}.

Decompose the above relation into BCNF using Relational Decomposition into BCNF with Nonadditive Join Property algorithm

Exercise -2

Consider the universal relation R = {A, B, C, D, E, F, G, H, I, J}

and the set of functional dependencies

G = {AB→ C, BD→ EF, AD→GH, A→ I, H→J }.

Decompose the above relation into BCNF using Relational Decomposition into BCNF with Nonadditive Join Property algorithm

3. Relational Synthesis into 3NF with Dependency Preservation and Nonadditive (Lossless) Join Property

- The Algorithm yields a decomposition D of R that does the following:
  1. Preserves dependencies
  2. Has the nonadditive join property
  3. Is such that each resulting relation schema in the decomposition is in 3NF

If we want a decomposition to have both nonadditive join property and to preserve dependencies, we have to be satisfied with relation schemas in 3NF rather than BCNF

Algorithm : **Relational Synthesis into 3NF with Dependency Preservation and Nonadditive (Lossless) Join Property**

Input: A universal relation R and a set of functional dependencies F on the attributes of R.

1. Find a minimal cover G for F
2. For each left-hand-side X of a functional dependency that appears in G create a relation schema in D with attributes $\{X \cup \{A_1\} \cup \{A_2\} \ldots \cup \{A_k\}\}$, where $X \rightarrow A_1$, $X \rightarrow A_2$, ... , $X \rightarrow A_k$ are the only dependencies in G with X as left· hand-side (**X is the key of this relation**).
3. If none of the relation schemas in D contains a key of R, then create one more relation schema in D that contains attributes that form a key of R.

Finding a key of R

Step 3 of Algorithm 11.4 involves identifying a key K of R. The following Algorithm can be used to identify a key K of R based on the set of given functional dependencies F.

Algorithm : Finding a Key K for R Given a set F of Functional Dependencies

Input: A universal 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}};
```

Finding a key of R

Exercise - 1

Consider the relation schema R(A,B,C,D,E) and the functional dependency set F.

F = { A→B, BC→E, ED→A}

Using the algorithm **find a key for R**

Finding a key of R

Exercise - 2

Consider the relation schema R(A,B,C,D,E,F) and the functional dependency set F.

F = { A→D, B→EF, AB→C}

Using the algorithm fi**nd a key for R**

Finding a key of R

Exercise - 3

Use the algorithm to find a key of relation R(A, B, C, D, E) given the following set F of functional dependencies

$F = \{AB \rightarrow C, CD \rightarrow E, DE \rightarrow B\}$

Relational Synthesis into 3NF with Dependency Preservation and Nonadditive (Lossless) Join Property

Exercise - 1

Consider the universal relation R = (A, B, C, D, E, F, G) and the set of functional dependencies

$$F = \{A \rightarrow BC,\ BC \rightarrow F,\ B \rightarrow DCE,\ C \rightarrow DEF,\ E \rightarrow FG\}.$$

Decompose R into 3NF relations using Relational Synthesis algorithm that preserves dependency and lossless join properties

Relational Synthesis into 3NF with Dependency Preservation and Nonadditive (Lossless) Join Property

Exercise - 2

Consider the universal relation R = (A, B, C, D, E, F, G, H, I,J) and the set F of functional dependencies

F = {AB→C, A→ DE, B→F, F → GH, D→IJ}.

Decompose R into 3NF relations using Relational Synthesis algorithm that preserves dependency and lossless join properties

Relational Synthesis into 3NF with Dependency Preservation and Nonadditive (Lossless) Join Property

Exercise -3

Consider the universal relation R = (A, B, C, D, E, F, G, H, I, J) and the set of functional dependencies

$G = \{AB \rightarrow C, BD \rightarrow EF, AD \rightarrow GH, A \rightarrow I, H \rightarrow J\}$.

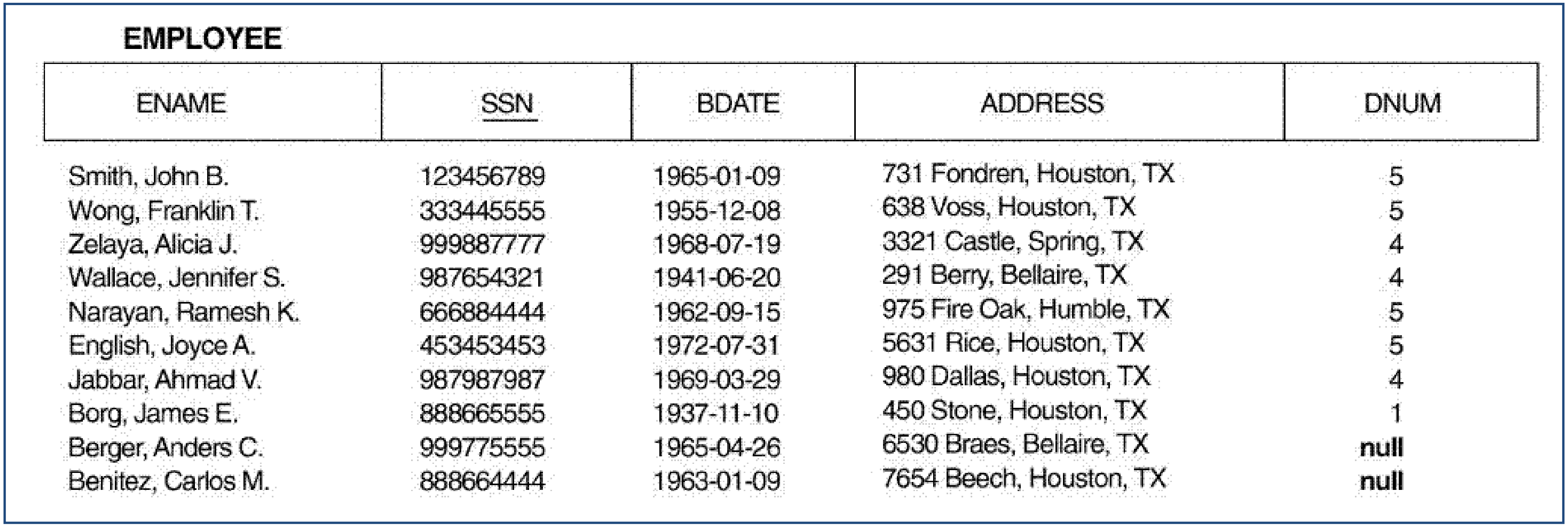
Decompose R into 3NF relations using Relational Synthesis algorithm that preserves dependency and lossless join properties

Problems with Null Values and Dangling Tuples

Null Values - Loss of Information

- Loss of Information occurs when some tuples have null values for attributes that will be used to join individual relations in the decomposition.

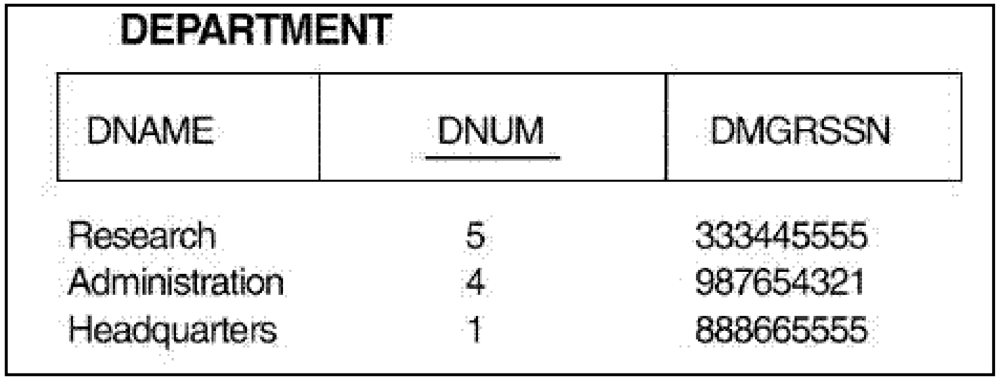
To illustrate this, consider two relations EMPLOYEE and DEPARTMENT



EMPLOYEE

ENAME	SSN	BDATE	ADDRESS	DNUM
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1
Berger, Anders C.	999775555	1965-04-26	6530 Braes, Bellaire, TX	**null**
Benitez, Carlos M.	888664444	1963-01-09	7654 Beech, Houston, TX	**null**

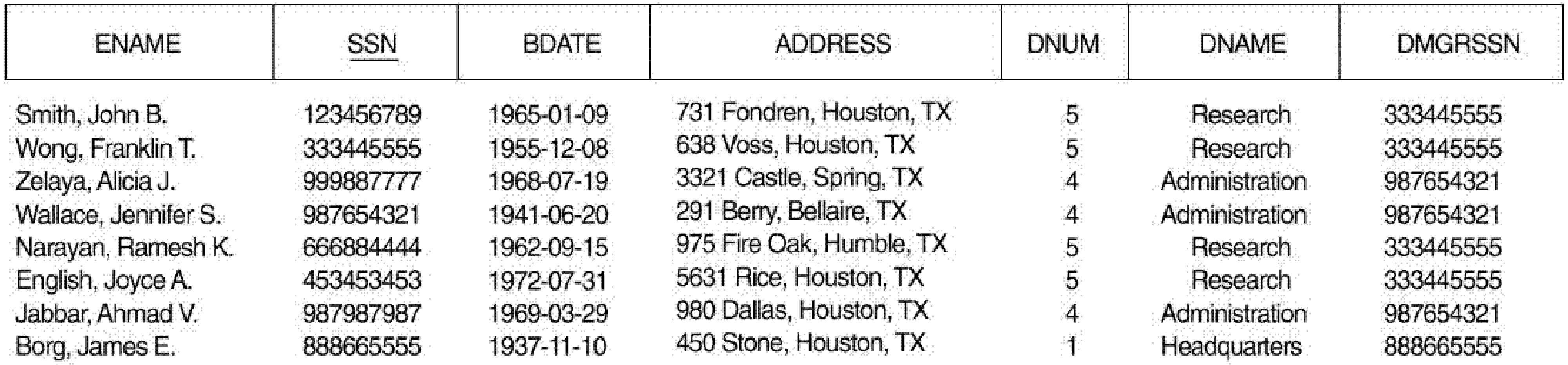
Null Values - Loss of Information



DEPARTMENT

DNAME	DNUM	DMGRSSN
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

EMPLOYEE * DEPARTMENT



ENAME	SSN	BDATE	ADDRESS	DNUM	DNAME	DMGRSSN
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5	Research	333445555
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5	Research	333445555
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4	Administration	987654321
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4	Administration	987654321
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5	Research	333445555
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5	Research	333445555
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4	Administration	987654321
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1	Headquarters	888665555

The last two tuples are lost in JOIN operation

Null Values - Loss of Information

Therefore ,

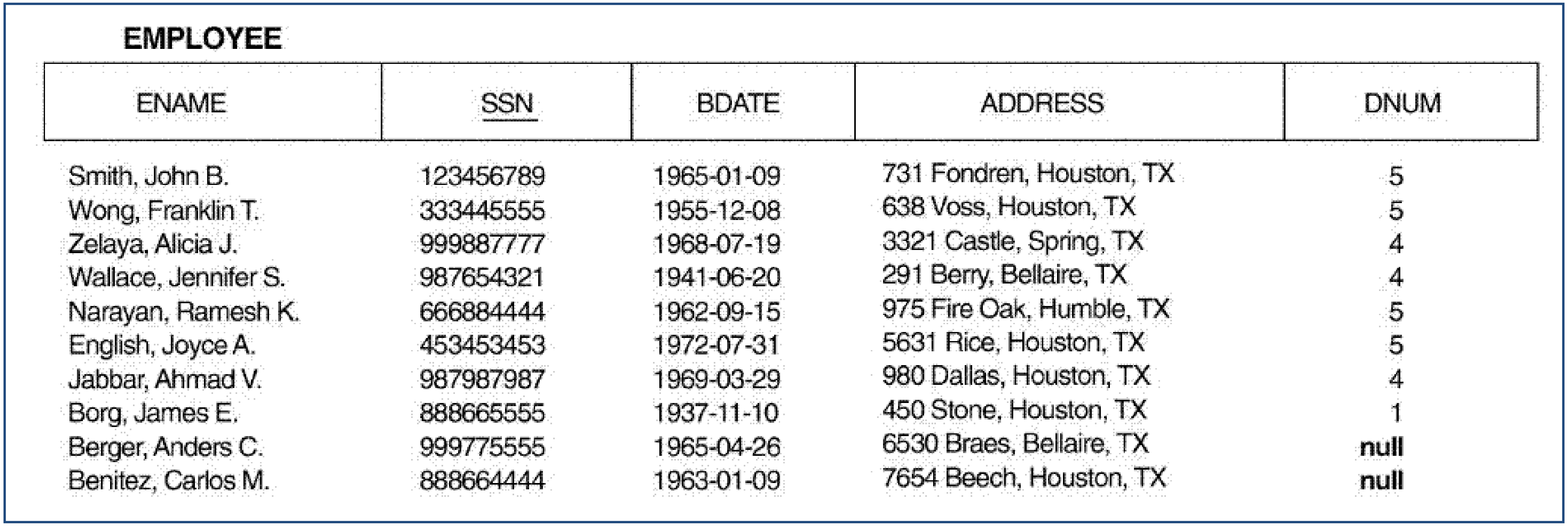
- Whenever a relational database schema is designed in which two or more relations are interrelated via foreign keys, particular care must be devoted to watching for potential null values in foreign keys.

- If nulls occur in other attributes, such as SALARY, their effect on built-in functions such as SUM and AVERAGE must be carefully evaluated.

Dangling tuples

A related problem is that of dangling tuples, which may occur if we carry a decomposition of the EMPLOYEE relation too far.

Suppose that we decompose the EMPLOYEE relation of Figure 11.2a further into EMPLOYEE_1 and EMPLOYEE_2



EMPLOYEE

ENAME	SSN	BDATE	ADDRESS	DNUM
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX	5
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX	5
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX	4
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX	4
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX	5
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX	5
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX	4
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX	1
Berger, Anders C.	999775555	1965-04-26	6530 Braes, Bellaire, TX	**null**
Benitez, Carlos M.	888664444	1963-01-09	7654 Beech, Houston, TX	**null**

Dangling tuples

(a) **EMPLOYEE_1**

ENAME	SSN	BDATE	ADDRESS
Smith, John B.	123456789	1965-01-09	731 Fondren, Houston, TX
Wong, Franklin T.	333445555	1955-12-08	638 Voss, Houston, TX
Zelaya, Alicia J.	999887777	1968-07-19	3321 Castle, Spring, TX
Wallace, Jennifer S.	987654321	1941-06-20	291 Berry, Bellaire, TX
Narayan, Ramesh K.	666884444	1962-09-15	975 Fire Oak, Humble, TX
English, Joyce A.	453453453	1972-07-31	5631 Rice, Houston, TX
Jabbar, Ahmad V.	987987987	1969-03-29	980 Dallas, Houston, TX
Borg, James E.	888665555	1937-11-10	450 Stone, Houston, TX
Berger, Anders C.	999775555	1965-04-26	6530 Braes, Bellaire, TX
Benitez, Carlos M.	888664444	1963-01-09	7654 Beech, Houston, TX

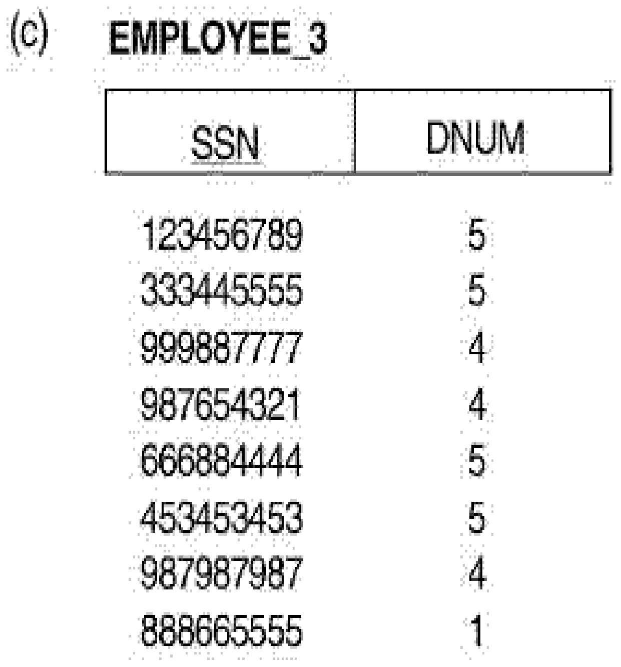
(b) **EMPLOYEE_2**

SSN	DNUM
123456789	5
333445555	5
999887777	4
987654321	4
666884444	5
453453453	5
987987987	4
888665555	1
999775555	null
888664444	null

In this decomposition , If apply EMPLOYEE_1 * EMPLOYEE_2 , we get the original relation.

Dangling tuples

Now suppose that we use the alternative representation for EMPLOYEE_2, where we do not include a tuple if the employee has not been assigned a department .



(c) EMPLOYEE_3

SSN	DNUM
123456789	5
333445555	5
999887777	4
987654321	4
666884444	5
453453453	5
987987987	4
888665555	1

Now in the result of EMPLOYEE_1 * EMPLOYEE_2, the last two tuples are not found. These are called ***dangling tuples*** because they are represented in only one of the two relations that represent employees

SUMMARY OF THE ALGORITHMS

ALGORITHM	INPUT	OUTPUT	PROPERTIES/PURPOSE	REMARKS
11.1	A decomposition D of R and a set F of functional dependencies	Boolean result: yes or no for nonadditive join property	Testing for nonadditive join decomposition	See a simpler test in Section 11.1.4 for binary decompositions
11.2	Set of functional dependencies F	A set of relations in 3NF	Dependency preservation	No guarantee of satisfying lossless join property
11.3	Set of functional dependencies F	A set of relations in BCNF	Nonadditive join decomposition	No guarantee of dependency preservation
11.4	Set of functional dependencies F	A set of relations in 3NF	Nonadditive join **AND** dependency-preserving decomposition	May not achieve BCNF
11.4a	Relation schema R with a set of functional dependencies F	Key K of R	To find a key K (that is a subset of R)	The entire relation R is always a default superkey

MULTIVALUED DEPENDENCIES AND FOURTH NORMAL FORM

MULTIVALUED DEPENDENCIES

- In many cases relations have constraints that cannot be specified as functional dependencies.

- If there are two or more multivalued independent attributes to repeat every value of one of the attributes with every value of the other attribute.

- Because we need to keep the relation state consistent and to maintain the independence

MULTIVALUED DEPENDENCIES

Example :

<u>ENAME</u>	<u>PNAME</u>	<u>DEPENDENT_NAME</u>
Smith	X	John
Smith	Y	Michael
Smith	X	Michael
Smith	Y	John

- Here to keep the relation state consistent, we must have a separate tuple to represent every combination of an employee's dependent and an employee's project.
- This constraint is specified as a **multivalued dependency**

Formal Definition

A **multivalued dependency (MVD)** $X \twoheadrightarrow Y$ 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:

- $t_3[X] = t_4[X] = t_1[X] = t_2[X]$.
- $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$.
- $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$.

Formal Definition

Note the Following :

1. Whenever X —>> Y holds, we say that ***X multidetermines Y***
2. X —>> Y implies X —>> Z, and therefore it is sometimes written as X —>> YI Z.
3. An MVD *X* —>> *Y* in *R* is called a **trivial MVD** if
   - *Y* is a subset of *X*, OR
   - *X* ∪ *Y* = *R*.
4. An MVD that satisfies neither (a) nor (b) is called a **nontrivial MVD**

Formal Definition

Example for trivial MVD

EMP_PROJECTS Relation

ENAME	PNAME
Smith	X
Smith	Y

ENAME —>> PNAME.

A trivial MVD will hold in *any relation state r of R*

Inference Rules for Functional and Multivalued Dependencies

The following inference rules IRI through *IRS form a sound and complete set for inferring* functional and multivalued dependencies from a given set of dependencies

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 $X \rightarrow Z$.

Inference Rules for Functional and Multivalued Dependencies

- IRI through IR3 are Armstrong's inference rules for FDs alone. IR4 through IR6 are inference rules pertaining to MVDs only.
- IR7 and *IR8 relate FDs and MVDs*
- IR7 says that a functional dependency is a *special case of a multivalued dependency : that is, every FD is also an MVD*

Inference Rules for Functional and Multivalued Dependencies

An FD *X -> Y* is an MVD *X-->> Y* with the *additional implicit restriction that at most one value of Y is associated with each value of X*

Given a set F of functional and multivalued dependencies specified on R = {A1, A2, ... , An}, we can use IR1 through IR8 to infer the (complete) set of all dependencies (functional or multivalued) P that will hold in every relation state r of R that satisfies F.

Fourth Normal (4NF)

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.

Note: F^+ is the (complete) set of all dependencies (functional or multivalued) that will hold in every relation state *r* of *R* that satisfies *F*. It is also called the **closure** of *F*.

Fourth Normal (4NF) - Example

(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

Fourth Normal (4NF) - Example

(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

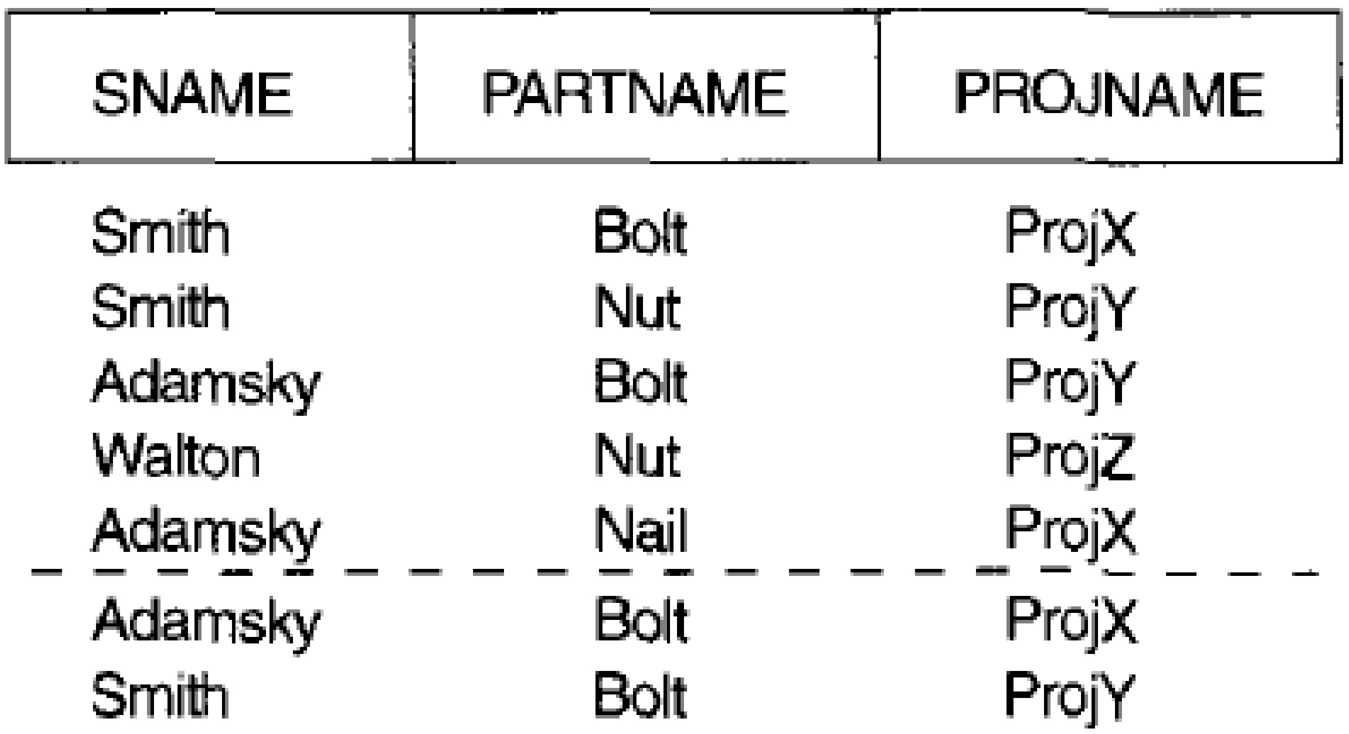
EMP_DEPENDENTS

ENAME	DNAME
Smith	Anna
Smith	John
Brown	Jim
Brown	Joan
Brown	Bob

Fourth Normal (4NF) - Example

The Supply relation is already in 4NF and should not be decomposed.

(c) **SUPPLY**



SNAME	PARTNAME	PROJNAME
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

Lossless (Nonadditive) Join Decomposition into 4NF Relations

PROPERTY LJ1'

The relation schemas R_1 and R_2 form a lossless (non-additive) join decomposition of R with respect to a set F of functional *and* multivalued dependencies if and only if

$(R_1 \cap R_2) \twoheadrightarrow (R_1 - R_2)$

or by symmetry, if and only if

$(R_1 \cap R_2) \twoheadrightarrow (R_2 - R_1)$).

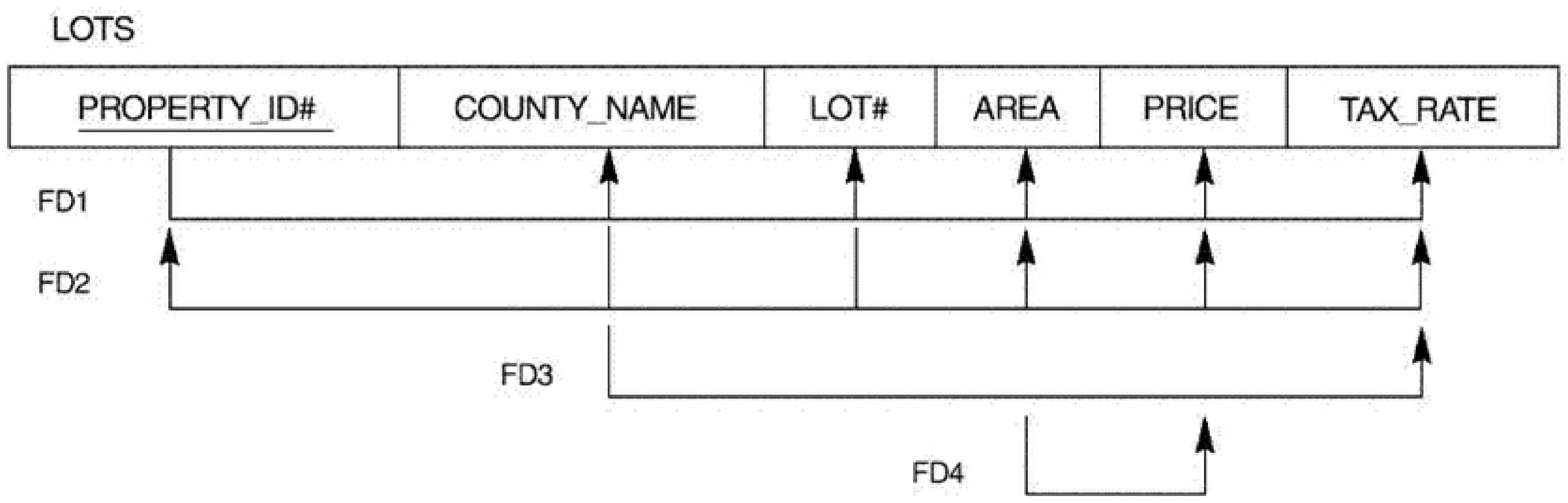
Algorithm : Relational decomposition into 4NF relations with non-additive join property

Input: A universal relation R and a set of functional and multivalued dependencies F.

```
1.  Set D := { R };
2.  While there is a relation schema $Q$ in $D$ that is not in 4NF do
    {  choose a relation schema $Q$ in $D$ that is not in 4NF;
       find a nontrivial MVD $X \twoheadrightarrow Y$ in $Q$ that violates 4NF;
       replace $Q$ in $D$ by two relation schemas $(Q - Y)$ and $(X \cup Y)$;
    };
```

Problems :

- In what normal form is the LOTS relation schema in Figure with respect to the restrictive interpretations of normal form that take only the *primary key* into account?
- Would it be in the same normal form if the general definitions of normal form were used?



Problems :

- Prove that any relation schema with two attributes is in BCNF

- Consider the universal relation R = {A, B, C, D, E, F, G, H, I} and the set of functional dependencies F = { AB -> C, A -> DE, B -> F, F -> G, H, D -> IJ }.

  i. What is the key for R?
  ii. Decompose R into 2NF, then 3NF relations.

- Repeat above exercise for the following different set of functional dependencies G = { {A, B} -> {C}, {B, D} -> {E, F}, {A, D} -> {G, H}, {A} -> {I}, {H} -> {J} }.

Problems :

- Consider a relation R(A,B,C,D,E) with the following dependencies:
  AB -> C
  CD -> E
  DE -> B

  Is AB a candidate key of this relation? If not, is ABD?
  Explain your answer.

Problems :

- Consider the following relation:

CAR_SALE(Car#, Date_sold, Salesman#, Commision%, Discount_amt)

- Assume that a car may be sold by multiple salesmen and hence {CAR#, SALESMAN#} is the primary key. Additional dependencies are:

  Date_sold ->Discount_amt

  Salesman ->commission

  Car→Date_sold

Based on the given primary key, is this relation in 1NF, 2NF, or 3NF? Why or why not? How would you successively normalize it completely?

Problems :

- Show that the relation schemas produced by Algorithm 11.2 are in 3NF.