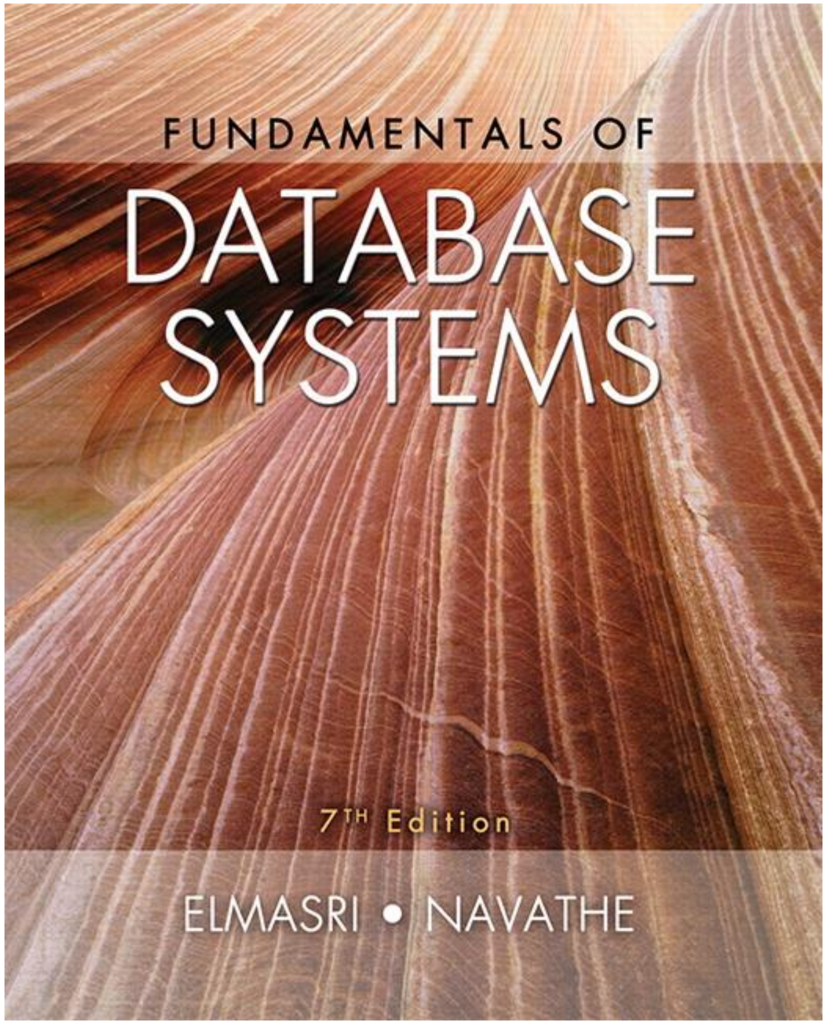
Lecture – 12th week

(Prof. Shamkant B Navathe)

November 7, 2023

sham@cc.gatech.edu

Relational Database Design Algorithms


FUNDAMENTALS OF
DATABASE SYSTEMS
7TH Edition
ELMASRI • NAVATHE

CHAPTER 15

Relational Database Design Algorithms and Further Dependencies

Chapter Outline

- 1. Further topics in Functional Dependencies
  - 1.1 Inference Rules for FDs
  - 1.2 Equivalence of Sets of FDs
  - 1.3 Minimal Sets of FDs
- 2. Properties of Relational Decompositions
- 3. Algorithms for Relational Database Schema Design
- 4. Nulls, Dangling Tuples, Alternative Relational Designs

We already covered Section 15.1 dealing with Armstrong's inference rules and studied closure, equivalence among sets of FDs, and computing minimal sets of FDs.

TODAY WE WILL DISCUSS THE NEXT THREE SECTIONS: 2,3,4.

Chapter Outline

- 5. Multivalued Dependencies and Fourth Normal Form – further discussion
- 6. Other Dependencies and Normal Forms
  - 6.1 Join Dependencies
  - 6.2 Inclusion Dependencies
  - 6.3 Dependencies based on Arithmetic Functions and Procedures
  - 6.2 Domain-Key Normal Form

THE ABOVE TOPICS ARE BEYOND OUR SCOPE AND WILL NOT BE COVERED. Will have some comments on 6.2 and 6.3 for your information.

DESIGNING A SET OF RELATIONS (1)

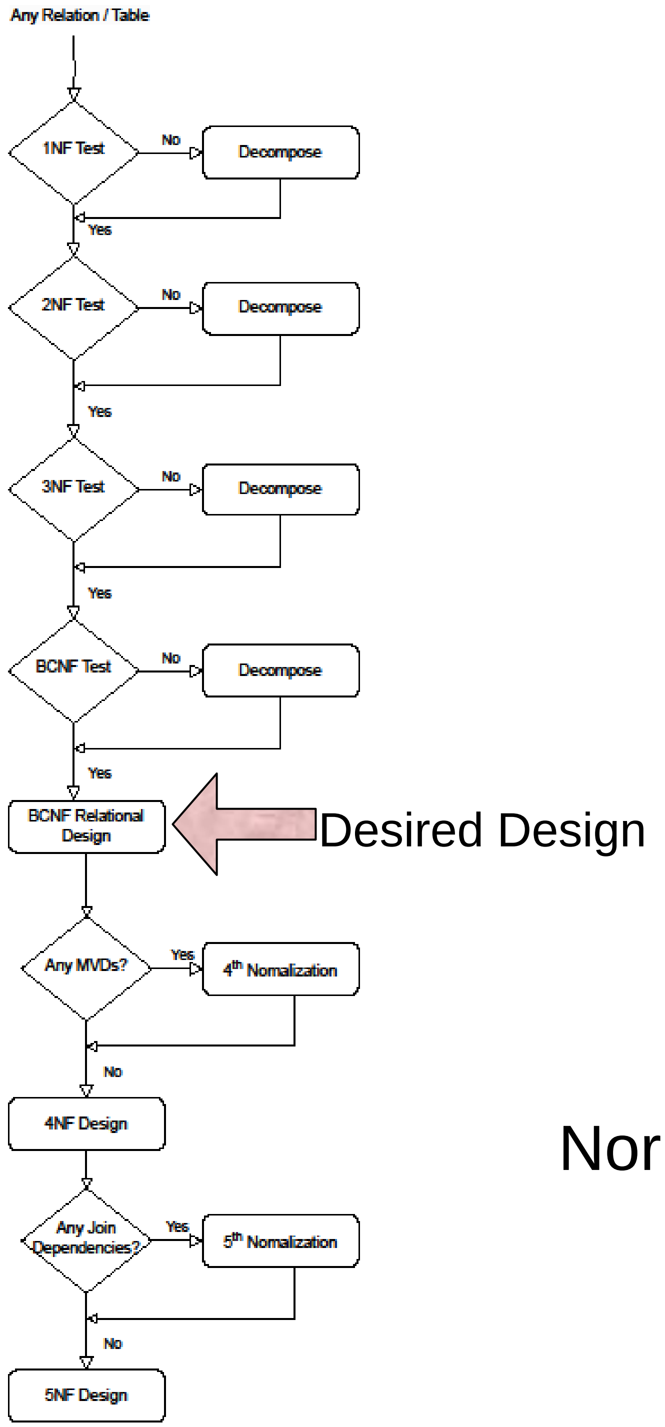
- **The Approach of Relational Synthesis (Bottom-up Design):**
  - Imagines all attributes are thrown into a single relation: called Universal relation
  - Assumes that all possible functional dependencies are known
  - First constructs a minimal set of FDs
  - Then applies algorithms that construct a target set of 3NF or BCNF relations.
  - Additional criteria may be needed to ensure the the *set of relations* in a relational database are satisfactory (see Algorithms 15.3, 15.4).

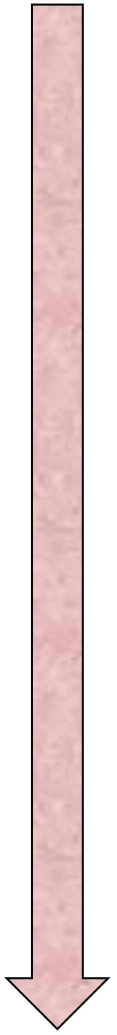
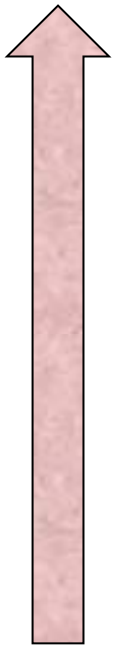
TOP-DOWN DESIGN

Also called “DESIGN by ANALYSIS”

- **Normalization:**
  - The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations
  - (may be considered as “improvement”, “purification” of a given design to make it better so that anomalies and redundancy are eliminated)
- **Normal form:**
  - Condition using keys and FDs of a relation to certify whether a relation schema is in a particular state of “goodness”.

The top-down design PROCESS



Normalization

Denormalization

Opposite Strategy: BOTTOM-UP DESIGN

Also known as “DESIGN by SYNTHESIS”

3NF and BCNF relations

⇧

MINIMUM COVER FOR THE UNIVERSAL RELATION

⇧

A UNIVERSAL RELATION

U (A1, A2, ……………………………………………..An)

DESIGNING A SET OF RELATIONS

- **BASIS:**
  - A PAPER PUBLISHED IN 1976 BY PHILLIP BERNSTEIN based on PhD work he did at Harvard:
  - **Title: Synthesizing third normal form relations from functional dependencies**
  - Journal: ACM Transactions on Database Systems
    - Volume 1, Issue 4, pp 277 –298 https://doi.org/10.1145/320493.320489

DESIGNING A SET OF RELATIONS

- **Goals:**
  - Lossless join property (a must)
    - Algorithm 15.3 tests for general losslessness of n-ary decompositions.
  - Dependency preservation property
    - Observe as much as possible
    - Algorithm 15.5 decomposes a relation into BCNF components by sacrificing the dependency preservation but guarantees losslessness.
  - Additional normal forms to remove unwanted dependencies- More of academic interest, less practically applicable
    - 4NF (based on multi-valued dependencies)
    - 5NF (based on join dependencies)

Algorithm to determine the key of a relation

- **Algorithm 15.2a 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;   key to be the WHOLE relation at first
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};
   }                    see if can eliminate attributes one by one
```

Deteremining the key of a relation

Example:

- **ORDER (order#, order_date,customer_id, amount, cust_phone#)**

1. Default key: entire relation

2. Start dropping attributes until you find combinations that uniquely determine each row in the table.

Note: application semantics/rules govern what constitutes a key.

customer_id, order_date ? : yes, but only if ….

cust_phone#, order_date ? : yes, but only if …..

Eventually: Order# ?

Other candidate keys may work if the rules for order processing support those keys

2. Properties of Relational Decompositions

- **Relation Decomposition and Insufficiency of Normal Forms:**
  - Universal Relation Schema:
    - A relation schema U = {A1, A2, …, An} that includes all the attributes of the database.
  - Universal relation assumption:
    - Every attribute name is unique.

Relational Decompositions with attribute preservation

2.1 Relation Decomposition and Insufficiency of Normal Forms (cont.):

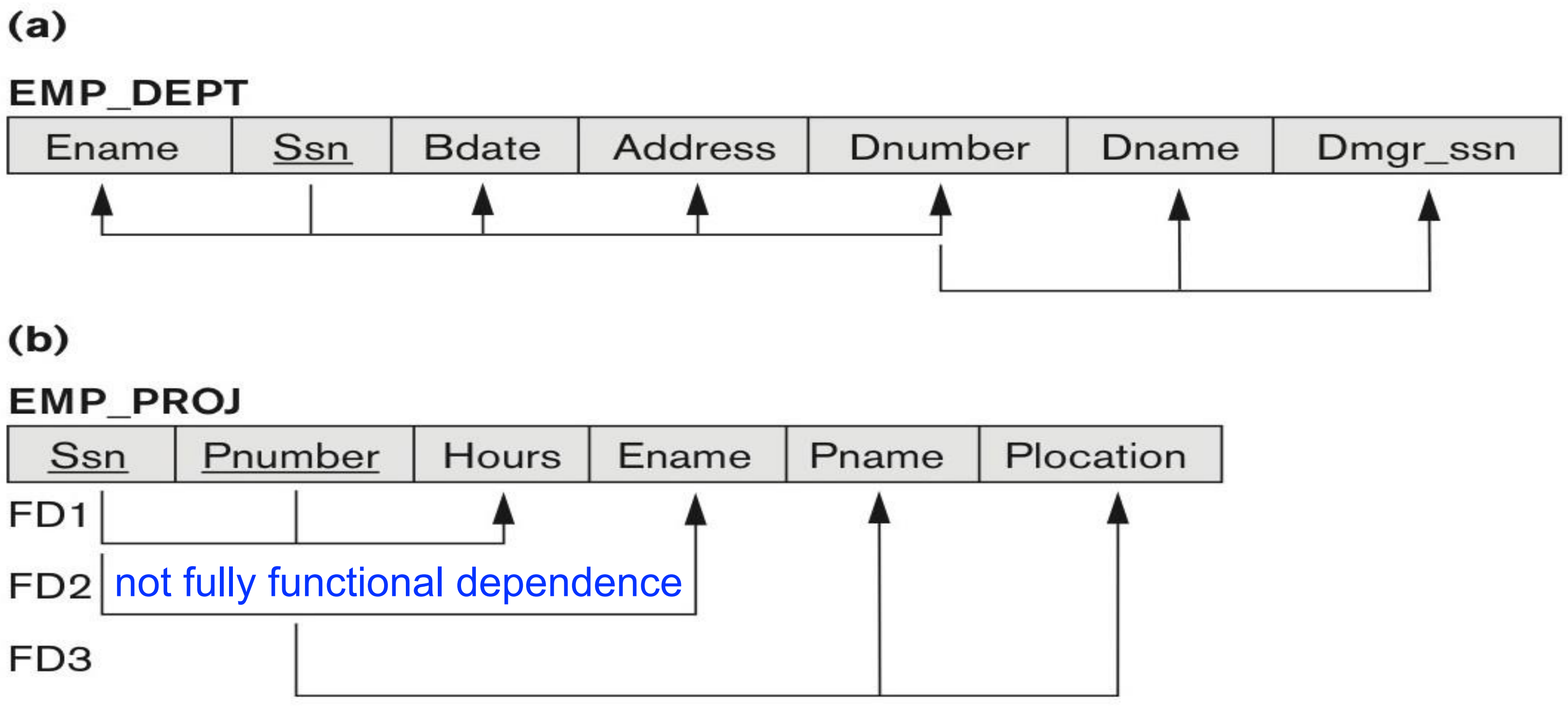
- Decomposition:
  - The process of decomposing the universal relation schema R into a set of relation schemas D = {R1,R2, …, Rm} that will become the relational database schema by using the functional dependencies.
- Attribute preservation condition:
  - Each attribute in R will appear in at least one relation schema R_i in the decomposition so that no attributes are “lost”.

Goals and Properties of Universal Relational Decomposition

- Another (eventual) goal of decomposition is to have each individual relation R_i in the decomposition D be in BCNF or 3NF.
- Additional properties of decomposition are needed to prevent from generating spurious tuples

Goals and Properties of Universal Relational Decomposition

Consider a poor example of design – we are given the EMP_PROJ relation (not in 2NF)



Goals and Properties of Universal Relational Decomposition

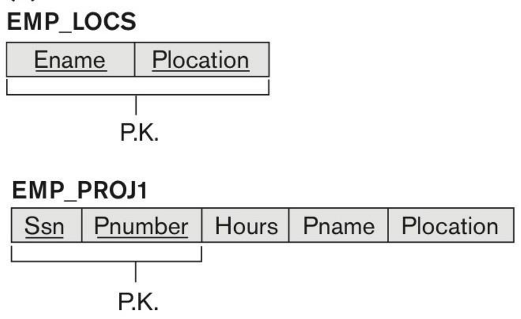
- Suppose EMP_PROJ is decomposed into two relations:

- EMP-LOCS (<u>Ename, Plocation</u>) in BCNF
- EMP_PROJ1 (<u>Ssn, Pno</u>, Hours, Pname, Plocation) in 1NF

so the question is is this a good decomposition?

Decomposing EMP_PROJ onto the relations EMP_LOCS and EMP_PROJ1- showing instances

(a)



(b)

EMP_LOCS

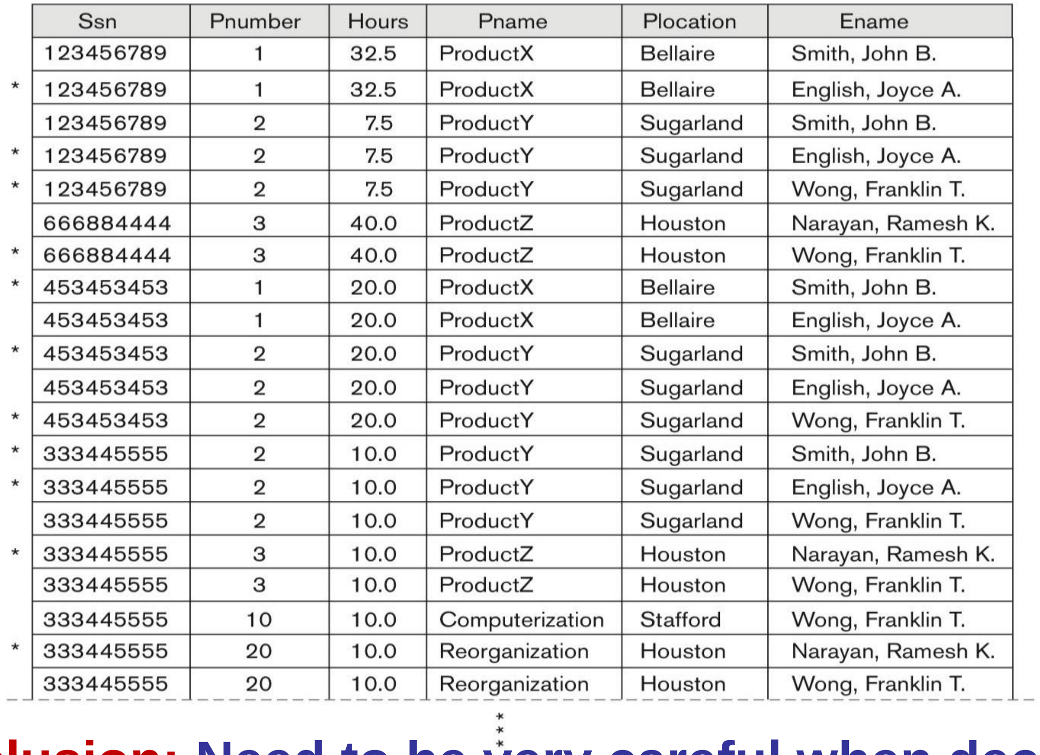
Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Sugarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Sugarland
Wong, Franklin T.	Sugarland
Wong, Franklin T.	Houston
Wong, Franklin T.	Stafford
Zelaya, Alicia J.	Stafford
Jabbar, Ahmad V.	Stafford
Wallace, Jennifer S.	Stafford
Wallace, Jennifer S.	Houston
Borg, James E.	Houston

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation
123456789	1	32.5	ProductX	Bellaire
123456789	2	7.5	ProductY	Sugarland
666884444	3	40.0	ProductZ	Houston
453453453	1	20.0	ProductX	Bellaire
453453453	2	20.0	ProductY	Sugarland
333445555	2	10.0	ProductY	Sugarland
333445555	3	10.0	ProductZ	Houston
333445555	10	10.0	Computerization	Stafford
333445555	20	10.0	Reorganization	Houston
999887777	30	30.0	Newbenefits	Stafford
999887777	10	10.0	Computerization	Stafford
987987987	10	35.0	Computerization	Stafford
987987987	30	5.0	Newbenefits	Stafford
987654321	30	20.0	Newbenefits	Stafford
987654321	20	15.0	Reorganization	Houston
888665555	20	NULL	Reorganization	Houston

Result of applying NATURAL JOIN to the tuples in EMP_PROJ1 and EMP_LOCS. Generated spurious tuples are marked by asterisks.

This was a bad decomposition that would result in a lot of spurious data after the natural join (on Plocation).
NOTE: This decomposition would fail the NJB test.



	Ssn	Pnumber	Hours	Pname	Plocation	Ename
	123456789	1	32.5	ProductX	Bellaire	Smith, John B.
*	123456789	1	32.5	ProductX	Bellaire	English, Joyce A.
	123456789	2	7.5	ProductY	Sugarland	Smith, John B.
*	123456789	2	7.5	ProductY	Sugarland	English, Joyce A.
*	123456789	2	7.5	ProductY	Sugarland	Wong, Franklin T.
	666884444	3	40.0	ProductZ	Houston	Narayan, Ramesh K.
*	666884444	3	40.0	ProductZ	Houston	Wong, Franklin T.
*	453453453	1	20.0	ProductX	Bellaire	Smith, John B.
	453453453	1	20.0	ProductX	Bellaire	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Smith, John B.
	453453453	2	20.0	ProductY	Sugarland	English, Joyce A.
*	453453453	2	20.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	2	10.0	ProductY	Sugarland	Smith, John B.
*	333445555	2	10.0	ProductY	Sugarland	English, Joyce A.
	333445555	2	10.0	ProductY	Sugarland	Wong, Franklin T.
*	333445555	3	10.0	ProductZ	Houston	Narayan, Ramesh K.
	333445555	3	10.0	ProductZ	Houston	Wong, Franklin T.
	333445555	10	10.0	Computerization	Stafford	Wong, Franklin T.
*	333445555	20	10.0	Reorganization	Houston	Narayan, Ramesh K.
	333445555	20	10.0	Reorganization	Houston	Wong, Franklin T.

*
*
*

Conclusion: Need to be very careful when decomposing relations during design.

Properties of Relational Decompositions

2.2 Dependency Preservation Property of a Decomposition:

- Definition: Given a set of dependencies F on R, the **projection** of F on R_i, denoted by π_{R1} (F) where R_i is a subset of R, is the set of dependencies $X \rightarrow Y$ in F^+ such that the attributes in X U Y are all contained in R_i.
- Hence, the projection of F on each relation schema R_i in the decomposition D is the set of functional dependencies in F^+, the closure of F, such that all their left- and right-hand-side attributes are in R_i.

Properties of Relational Decompositions

- **Dependency Preservation Property of a Decomposition (cont.):**
  - Dependency Preservation Property:
    - 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 Ri in D is equivalent to F; that is

$$((\pi_{R1}(F)) \cup \ldots \cup (\pi_{Rm}(F)))^+ = F^+$$

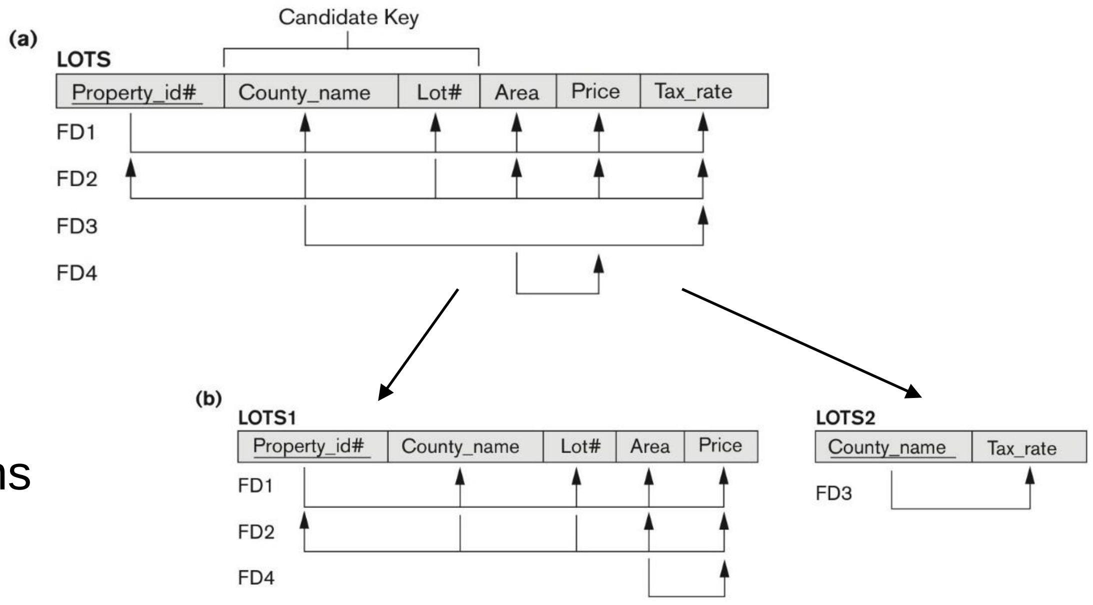
- Claim 1:
  - It is always possible to find a dependency-preserving decomposition D with respect to F such that each relation R_i in D is in 3NF.

Properties of Relational Decompositions

- **Example: Fig. 14.12 – Second Normalization**

Relation LOTS

Decomposed Relations LOTS1 and LOTS2



- **The decomposition preserved all FDs: FD1, FD2, FD3, FD4.**
- **Is it non-additive (lossless)? – Apply the NJB test.**

Relational Decomposition: Non-additive Join Binary (NJB) Property

- **PROPERTY NJB (non-additive join test for binary decompositions):** A decomposition D = {R1, R2} of R has the lossless join property with respect to a set of functional dependencies F on R *if and only if* either
  - The f.d. ((R1 ∩ R2) → (R1- R2)) is in F^+, or
  - The f.d. ((R1 ∩ R2) → (R2 - R1)) is in F^+.

LOTS (Propertyid#, County_name, Lot#, Area, Price, Tax_rate)

Is decomposed into:

- R1: LOTS1 (Propertyid#, County_name, Lot#, Area, Price)
- R2: LO TS2 (County_name, Tax_rate)

(R1 ∩ R2) = County_name

R2 - R1 is Tax_rate.

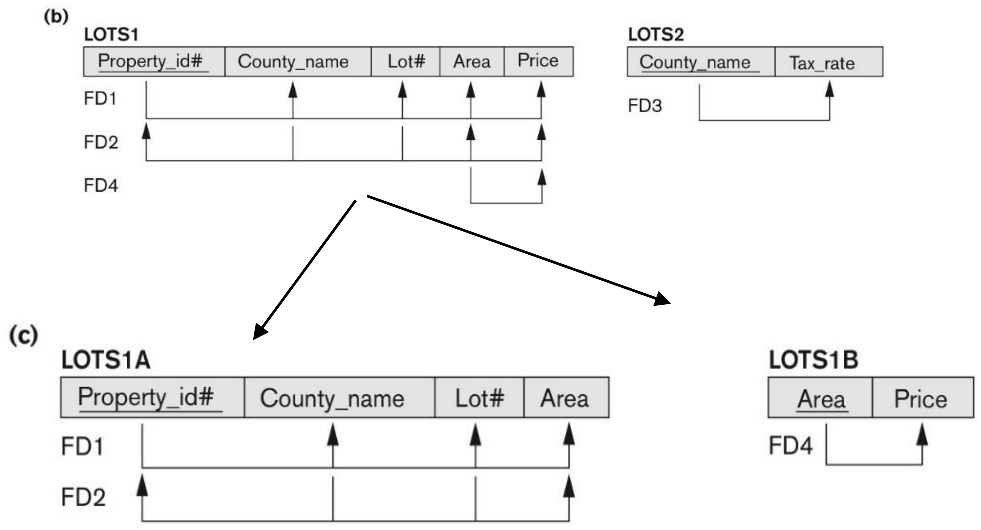
Since County_name → Tax_rate, the decomposition is good!

Properties of Relational Decompositions

- **Example: Fig. 14.12 – Third Normalization**

Decomposed Relations LOTS1 and LOTS2
In 2NF

Further decomposition of LOTS1 into LOTS1A and LOTS1B
in 3NF



- **This decomposition preserved all FDs of LOTS1, namely, : FD1, FD2, FD4.**
- **Is it non-additive (lossless)? – Apply the NJB test.**

Properties of Relational Decompositions

REMEMBER THE TEST AGAIN:

- The f.d. ((R1 ∩ R2) → (R1- R2)) is in F^+, or
- The f.d. ((R1 ∩ R2) → (R2 - R1)) is in F^+.

LOTS1 (Propertyid#, County_name, Lot#, Area, Price)

Is decomposed into:

- R1: LOTS1A (Propertyid#, County_name, Lot#, Area)
- R2: LO TS1B (Area, Price)

(R1 ∩ R2) = Area

R2 - R1 is Price.

Since Area→ Price, the decomposition is <u>good</u>!

You will notice that all the decompositions we showed for second and third normalization (during last lecture on Normalization) satisfy the NJB property and hence are good. Check that they also preserved the FDs.

Relational Decomposition: General Definition of Non-additive Lossless Join Property

2.3 Non-additive (Lossless) Join Property of a Decomposition:

- Definition: Lossless join property: 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:

$$* (\pi_{R1}(r), \ldots, \pi_{Rm}(r)) = r$$

- Note: The word loss in lossless refers to loss of information, not to loss of tuples. In fact, for “loss of information” a better term is “**addition of spurious information**”

Relational Decomposition : Algorithm to test non-additive join property for n-ary decomposition

Lossless (Non-additive) Join Property of a Decomposition :

- **Algorithm 15.3: Testing for Lossless Join Property**
  - **Input**: A universal relation R, a decomposition D = {R1, R2, ..., Rm} of R, and a set F of functional dependencies.

```
1. Create an initial matrix S with one row i for each relation Ri in D, and
   one column j for each attribute Aj in R.
2. Set S(i,j):=bij for all matrix entries. (* each bij is a distinct symbol
   associated with indices (i,j) *).
3. For each row i representing relation schema Ri
        {for each column j representing attribute Aj
            {if (relation Ri includes attribute Aj) then set S(i,j):= aj;};};
```

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

- CONTINUED on NEXT SLIDE

Relational Decomposition : Algorithm to test non-additive join property for n-ary decomposition (contd.)

- **Lossless (Non-additive) Join Property of a Decomposition (cont.):**

Algorithm 15.3: Testing for Lossless Join Property (continued)

```
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 which 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 the 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 appear 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.
```

Examples of n-ary decomposition testing

Figure 15.1 Nonadditive join test for n-ary decompositions.
(a) Case 1: Decomposition of EMP_PROJ into EMP_PROJ1 and EMP_LOCS fails test. Hence, not a good decomposition.
(b) A decomposition of EMP_PROJ that has the lossless join property.

(a)

$R = \{\text{Ssn, Ename, Pnumber, Pname, Plocation, Hours}\}$ $\qquad D = \{R_1, R_2\}$

$R_1 = \text{EMP\_LOCS} = \{\text{Ename, Plocation}\}$

$R_2 = \text{EMP\_PROJ1} = \{\text{Ssn, Pnumber, Hours, Pname, Plocation}\}$

$F = \{\text{Ssn} \rightarrow \text{Ename; Pnumber} \rightarrow \{\text{Pname, Plocation}\}; \{\text{Ssn, Pnumber}\} \rightarrow \text{Hours}\}$

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	b_{11}	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)

Now consider a "proper" decomposition:

(b)

EMP

Ssn	Ename

PROJECT

Pnumber	Pname	Plocation

WORKS_ON

Ssn	Pnumber	Hours

Examples of n-ary decomposition testing (contd.)

Nonadditive join test for n-ary decompositions. *(Figure 15.1)*
(c) Case 2: Decomposition of EMP_PROJ into EMP, PROJECT, and WORKS_ON satisfies test.

(c) $R = \{\text{Ssn, Ename, Pnumber, Pname, Plocation, Hours}\}$ $D = \{R_1, R_2, R_3\}$
$R_1 = \text{EMP} = \{\text{Ssn, Ename}\}$
$R_2 = \text{PROJ} = \{\text{Pnumber, Pname, Plocation}\}$
$R_3 = \text{WORKS\_ON} = \{\text{Ssn, Pnumber, Hours}\}$

$F = \{\text{Ssn} \rightarrow \text{Ename; Pnumber} \rightarrow \{\text{Pname, Plocation}\}; \{\text{Ssn, Pnumber}\} \rightarrow \text{Hours}\}$

	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)

	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_2	a_3	~~b_{34}~~ a_4	~~b_{35}~~ a_5	a_6

making a row all A's

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

(1) Because Ssn → Ename and both these have a's in row1, the a1 in row3 causes Ename to be set to a2. (2) Because Pnumber → (Pname, Plocation) and all these columns have an a in row2, the a3 in row3 causes a4, a5 to be set in row3.

Test for checking non-additivity of Binary Relational Decompositions (NJB TEST)

2.4 Testing Binary Decompositions for Non-additive Join (Lossless Join) Property

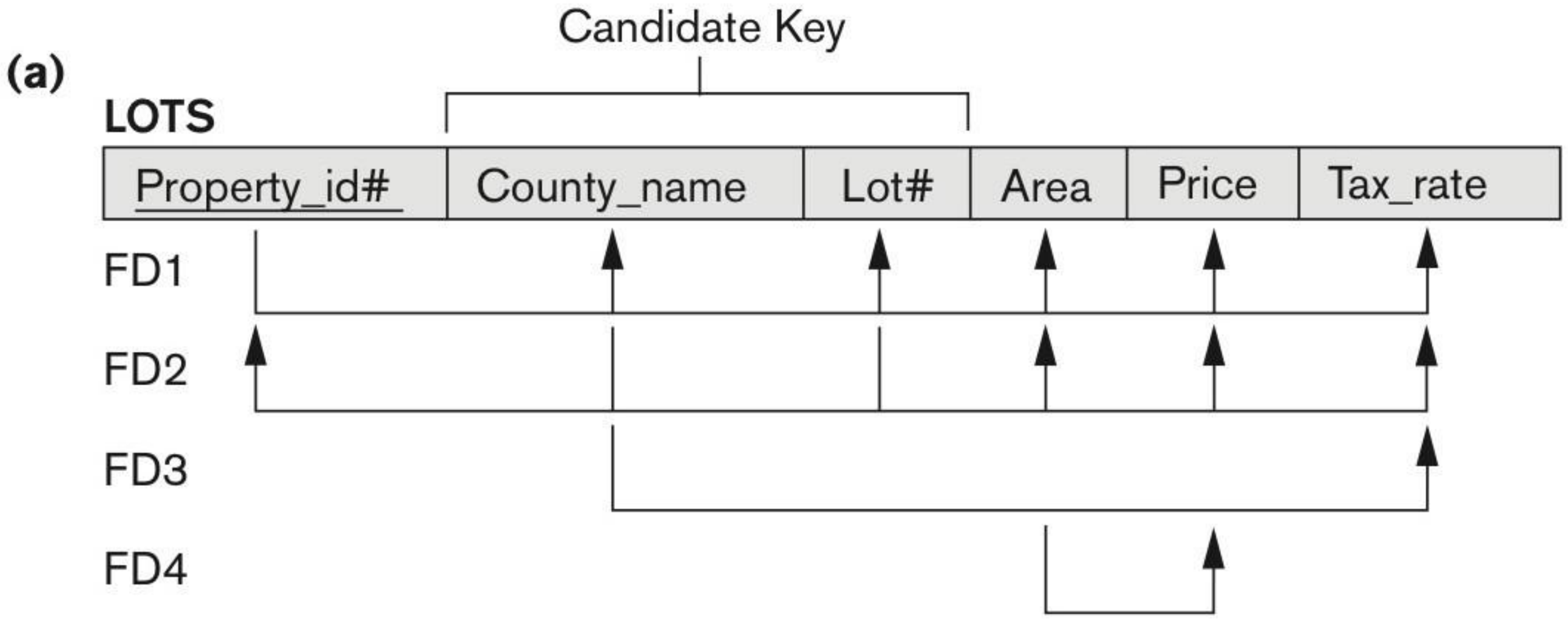
- **Binary Decomposition:** Decomposition of a relation R into two relations.
- **PROPERTY NJB (non-additive join test for binary decompositions):** A decomposition D = {R1, R2} of R has the lossless join property with respect to a set of functional dependencies F on R *if and only if* either
  - The f.d. ((R1 ∩ R2) → (R1- R2)) is in F^+, or
  - The f.d. ((R1 ∩ R2) → (R2 - R1)) is in F^+.

Properties of Relational Decompositions

2.5 Successive Non-additive Join Decomposition:

- **Claim 2 (Preservation of non-additivity in successive decompositions):**
  - If a decomposition D = {R1, R2, ..., Rm} of R has the lossless (non-additive) join property with respect to a set of functional dependencies F on R,
  - and if a decomposition Di = {Q1, Q2, ..., Qk} of Ri has the lossless (non-additive) join property with respect to the projection of F on Ri,
    - then the decomposition D2 = {R1, R2, ..., Ri-1, Q1, Q2, ..., Qk, Ri+1, ..., Rm} of R has the non-additive join property with respect to F.

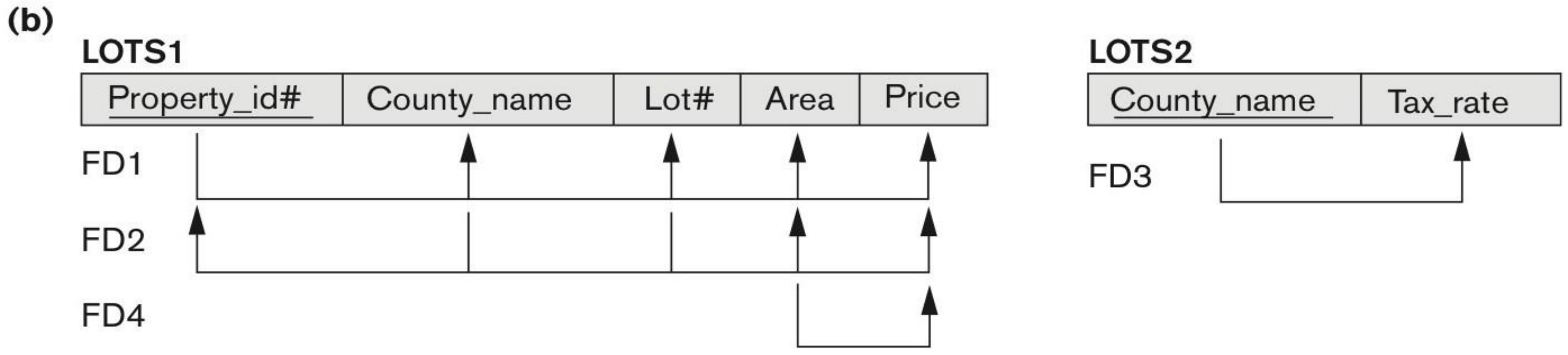
Example: The LOTS relation with its functional dependencies FD1 through FD4



continued on next slide

Normalization of LOTS into 2NF : Decomposing LOTS into the 2NF relations LOTS1 and LOTS2

We showed this decomposition is non-additive : LOTS into LOTS1 and LOTS2

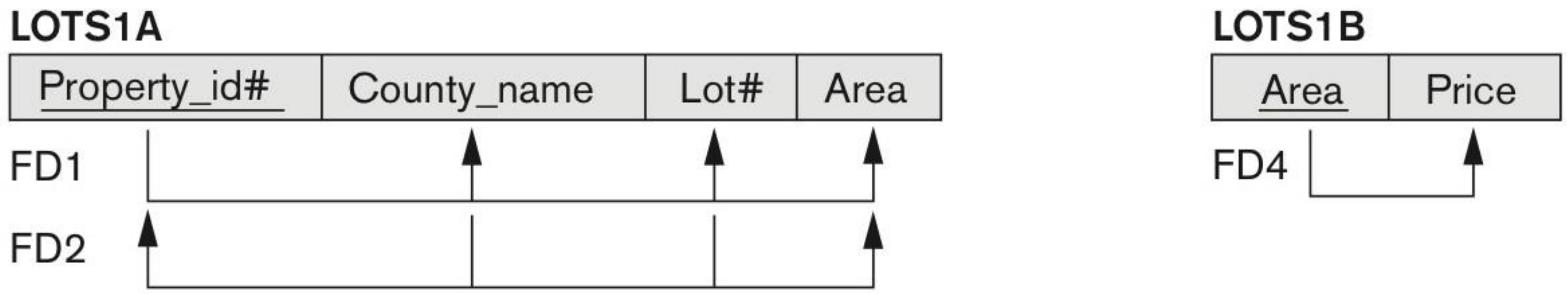


continued on next slide

Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B

We showed this decomposition is non-additive as well : LOTS1 into LOTS1A and LOTS1B.

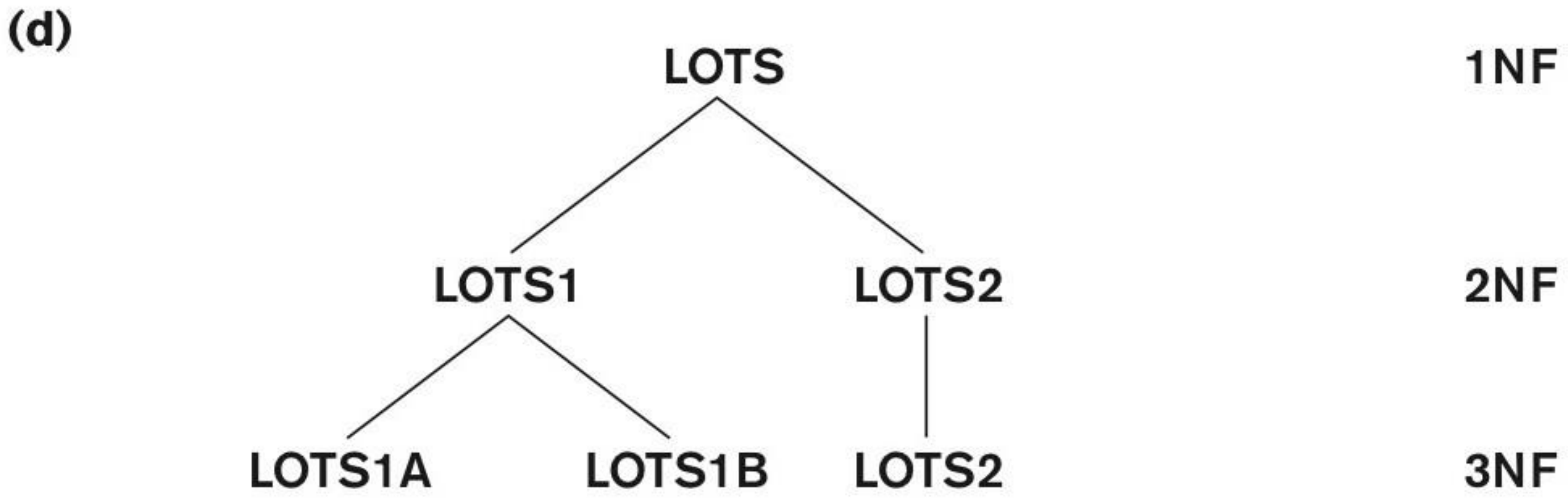
(c)



continued on next slide

Progressive normalization of LOTS into a 3NF design- preserves non-additivity property

Hence, by applying the successive decomposition non-additivity property, we claim that the decomposition of LOTS into LOTS1A, LOTS1B and LOTS2 is non-additive



3. Algorithms for Relational Database Schema Design (1)

- **Design of 3NF Schemas:**

Algorithm 15.4 Relational Synthesis into 3NF with Dependency Preservation and Non-Additive (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 (use Algorithm 15.2).

2. For each left-hand-side X of a functional dependency that appears in G,

create a relation schema in D with attributes {X U {A1} U{A2} ... U {Ak}},

where X → A1, X → A2, ..., X –>Ak are the only dependencies in G with X as left-hand-side (X is the key of this relation).

3. Algorithms for Relational Database Schema Design (2)

- **Design of 3NF Schemas: (contd.)**

Algorithm 15.4 Relational Synthesis into 3NF with Dependency Preservation and Non-Additive (Lossless) Join Property

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

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. *(Use Algorithm 15.4a to find the key of R)*

4. Eliminate redundant relations from the resulting set of relations in the relational database schema. A relation R is considered redundant if R is a projection of another relation S in the schema; alternately, R is subsumed by S.

Algorithms for Relational Database Schema Design –EXAMPLE of relational synthesis (1)

- **Example 1 of Algorithm 15.4.**

Consider the following universal relation:

U (Emp_ssn, Pno, Esal, Ephone, Dno, Pname, Plocation)

- Emp_ssn, Esal, and Ephone refer to the Social Security number, salary, and phone number of the employee.

- Pno, Pname, and Plocation refer to the number, name, and location of the project.

- Dno is the department number.

- The following dependencies are present:
- FD1: Emp_ssn $\rightarrow$ {Esal, Ephone, Dno}
- FD2: Pno $\rightarrow$ { Pname, Plocation}
- FD3: Emp_ssn, Pno $\rightarrow$ {Esal, Ephone, Dno, Pname, Plocation}

Algorithms for Relational Database Schema Design –EXAMPLE of relational synthesis (2)

- **Example 1 of Algorithm 15.4. –contd.**
- By virtue of FD3, the attribute set {Emp_ssn, Pno} represents a key of the universal
- relation.
- Hence $\mathcal{F}$, the set of given FDs, includes

{Emp_ssn → Esal, Ephone, Dno;
Pno → Pname, Plocation;
Emp_ssn, Pno → Esal, Ephone, Dno, Pname, Plocation}.

- To determine min cover of $\mathcal{F}$, we note that FD3 can be expressed as (In Step2 of Algo 15.2)

Emp_ssn, Pno → Esal
Emp_ssn, Pno → Ephone
Emp_ssn, Pno → Dno splitting up the FDs
Emp_ssn, Pno → Pname and
Emp_ssn, Pno → Plocation
In Step 3 of Algo 15.2 - the first three FDs above, Pno is extraneous and in the last two, Emp_ssn is extraneous. everything in the last FD is extraeous

Algorithms for Relational Database Schema Design –EXAMPLE of relational synthesis (3)

- **Example 1 of Algorithm 15.4. –contd.**
- Algorithm 15.4: second step produces relations *R*1 and *R*2 as:

*R*1 (Emp_ssn, Esal, Ephone, Dno)

*R*2 (Pno, Pname, Plocation)

- Algorithm 15.4: third step - we generate a relation corresponding to the key {Emp_ssn, Pno} of U

- Hence, the resulting design contains:

*R*1 (<u>Emp_ssn</u>, Esal, Ephone, Dno)

*R*2 (<u>Pno</u>, Pname, Plocation)

*R*3 (<u>Emp_ssn, Pno</u>)

We can easily see that the final design meets 3NF

Algorithms for Relational Database Schema Design –EXAMPLE of relational synthesis (4)

- **Example 2 of Algorithm 15.4.**
- Consider the relation schema LOTS1A
- Assume that this relation is given as a universal relation :

U (Property_id, County, Lot#, Area) with the following functional dependencies:

- *FD1: Property_id → Lot#, County, Area*
- *FD2: Lot#, County → Area, Property_id*
- *FD3: Area → County*

Represent the functional dependencies as the set in an abbreviated form

F: { P → LCA, LC → AP, A → C }

If we apply the

minimal cover Algorithm 15.2 to F, (in step 2) we first represent the set F as

F: :{P → L, P → C, P → A, LC → A, LC → P, A → C}

In the set F: , P → A can be inferred from P → LC and LC → A; hence P → A by transitivity and is therefore redundant. Hence, Algorithm 15.2 (in step 4) removes this redundant FD from the set F:

Algorithms for Relational Database Schema Design –EXAMPLE of relational synthesis (5)

- **Example 2 of Algorithm 15.4 – contd.**
- Thus, one possible minimal cover is
- Minimal cover FX: {P → LC, LC → A, LC → P, A → C}
- Algorithm 15.4 - step 2 will produce design X using the above minimal cover FX as

Design X: R1 (P, L, C), R2 (L, C, A, P), and R3 (A, C)
Now step 4 of Algorithm 15.4 applies: To reiterate:

Step 4 of Algorithm 15.4: Eliminate redundant relations from the resulting set of relations in the relational database schema. A relation R is considered redundant if R is a projection of another relation S in the schema; alternately, R is subsumed by S.

- we find that R3 is subsumed by R2
- we find that R1 is also subsumed by R2
- Hence both of those relations R1 and R3 are redundant. Thus the 3NF schema that achieves both of the desirable properties is (after removing redundant relations), is:
- **FINAL 3NF Design X**: R2 (L, C, A, P) which is same as the universal relation we started with; in other words it is identical to the relation
- LOTS1A (Property_id, Lot#, County, Area) that we had determined to be in 3NF in Section 14.4.2.

Algorithms for Relational Database Schema Design –EXAMPLE of relational synthesis (6)

- **Example 2 of Algorithm 15.4 – contd.**

FINAL 3NF Design X: R2 (L, C, A, P) which is **same as the universal relation** we started with.

- In other words it is identical to the relation

LOTS1A (Property_id, Lot#, County, Area) that we had determined to be in 3NF in Section 14.4.2.

- Note: In the textbook we discuss an alternate min cover of F that leads to an alternate 3NF Design Y:

Design *Y*: *S*1 (P, A, L), *S*2 (L, C, P), and *S*3 (A, C)

(see pages 521-522 in the book for further details on how we arrived at this alternate 3NF design)

Algorithms for Relational Database Schema Design – Design of BCNF schemas

Design of BCNF Schemas

Algorithm 15.5: Relational Decomposition into BCNF with Lossless (non-additive) 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 υ Y);
   };
```

Assumption: No null values are allowed for the join attributes.

Algorithms for Relational Database Schema Design – Example of BCNF design

- **Design of BCNF Schemas**

Consider a simple relation for a High class restaurant:

CUST_TABLE (Cust#, Table#, Date, Waiter#, Bill_amount)

The attributes are self-explanatory

The FDs are:

Fd1: (Cust#, Table#, Date) → Waiter#, Bill_amount

Fd2 : Waiter# → Table# (a fancy restaurant where a waiter waits on a single table!)

The relation CUST_TABLE is in 3NF because if you apply the generalized definition of 3NF, Fd1 has LHS as superkey and Fd2 has RHS which is a prime attribute. However Fd 2 violates BCNF. Using Algorithm 15.5, … X → Y violates BCNF ……. where X is Waiter# and Y is Table#.

Hence the BCNF design is:

CUST_TABLE 1 (Cust#, Date, Waiter#, Bill_amount) and

WAITER (Waiter#, Table#)

Note that this meets non-additive decomposition property but loses Fd1.

4. Problems with Null Values and Dangling Tuples (1)

4.1 Problems with NULL values

- when some tuples have NULL values for attributes that will be used to join individual relations in the decomposition that may lead to incomplete results.
- E.g., see Figure 15.2(a), where two relations EMPLOYEE and DEPARTMENT are shown. The last two employee tuples—‘Berger’ and ‘Benitez’—represent newly hired employees who have not yet been assigned to a department
- If we want to retrieve a list of (Ename, Dname) values for all the employees. If we apply the NATURAL JOIN operation on EMPLOYEE and DEPARTMENT (Figure 15.2(b)), the two aforementioned tuples will *not* appear in the result.
- In such cases, LEFT OUTER JOIN may be used. The result is shown in Figure 15.2 (c).

.

Problems with Null Values and Dangling Tuples (2)

(a)

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

DEPARTMENT

Dname	Dnum	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

Figure 15.2
Issues with NULL-value joins. (a) Some EMPLOYEE tuples have NULL for the join attribute Dnum.

Problems with Null Values and Dangling Tuples (3)

natural join: NULL results not included

(b)

Ename	Ssn	Bdate	Address	Dnum	Dname	Dmgr_ssn
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

(c)

Ename	Ssn	Bdate	Address	Dnum	Dname	Dmgr_ssn
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
Berger, Anders C.	999775555	1965-04-26	6530 Braes, Bellaire, TX	NULL	NULL	NULL
Benitez, Carlos M.	888665555	1963-01-09	7654 Beech, Houston, TX	NULL	NULL	NULL

Left outer join to get correct results

Figure 15.2
Issues with NULL-value joins.
(b) Result of applying NATURAL JOIN to the EMPLOYEE and DEPARTMENT relations.
(c) Result of applying LEFT OUTER JOIN to EMPLOYEE and DEPARTMENT

Problems with Null Values and Dangling Tuples (4)

Problems with Dangling Tuples

- Consider the decomposition of EMPLOYEE into EMPLOYEE_1 and EMPLOYEE_2 as shown in Figure 15.3 (a) and !5.3 (b).
- Their NATURAL JOIN yields the original relation EMPLOYEE in Figure 15.2(a).
- We may use the alternative representation, shown in Figure 15.3(c), where we *do not include a tuple* in EMPLOYEE_3 if the employee has not been assigned a department (instead of including a tuple with NULL for Dnum as in EMPLOYEE_2).
- If we use EMPLOYEE_3 instead of EMPLOYEE_2 and apply a NATURAL JOIN on EMPLOYEE_1 and EMPLOYEE_3, the tuples for Berger and Benitez will not appear in the result; these are called **dangling tuples** in EMPLOYEE*.*

Problems with Null Values and Dangling Tuples (5)

(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.	888665555	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

(c) EMPLOYEE_3

Ssn	Dnum
123456789	5
333445555	5
999887777	4
987654321	4
666884444	5
453453453	5
987987987	4
888665555	1

Figure 15.3
The dangling tuple problem. (a) The relation EMPLOYEE_1 (includes all attributes of EMPLOYEE from Figure 15.2(a) except Dnum). (b) The relation EMPLOYEE_2 (includes Dnum attribute with NULL values). (c) The relation EMPLOYEE_3 (includes Dnum attribute but does not include tuples for which Dnum has NULL values).

About Normalization Algorithms

4.2 Discussion of Normalization Algorithms:

- Problems :
  - The database designer must first specify *all* the relevant functional dependencies among the database attributes. not possible in practise
  - These algorithms are *not deterministic* in general.
  - It is not always possible to find a decomposition into relation schemas that preserves dependencies and allows each relation schema in the decomposition to be in BCNF (instead of 3NF as in Algorithm 15.5).

Summary of Algorithms for Relational Database Schema Design (1)

Table 15.1 Summary of the Algorithms Discussed in This Chapter

Algorithm	Input	Output	Properties/Purpose	Remarks
15.1	An attribute or a set of attributes *X*, and a set of FDs *F*	A set of attributes in the closure of *X* with respect to *F* closure	Determine all the attributes that can be functionally determined from *X*	The closure of a key is the entire relation
15.2	A set of functional dependencies *F*	The minimal cover of functional dependencies minimal cover	To determine the minimal cover of a set of dependencies *F*	Multiple minimal covers may exist—depends on the order of selecting functional dependencies
15.2a	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
15.3	A decomposition *D* of *R* and a set *F* of functional dependencies	Boolean result: yes or no for nonadditive join property nonadditive	Testing for nonadditive join decomposition	See a simpler test NJB in Section 14.5 for binary decompositions

Summary of Algorithms for Relational Database Schema Design (2)

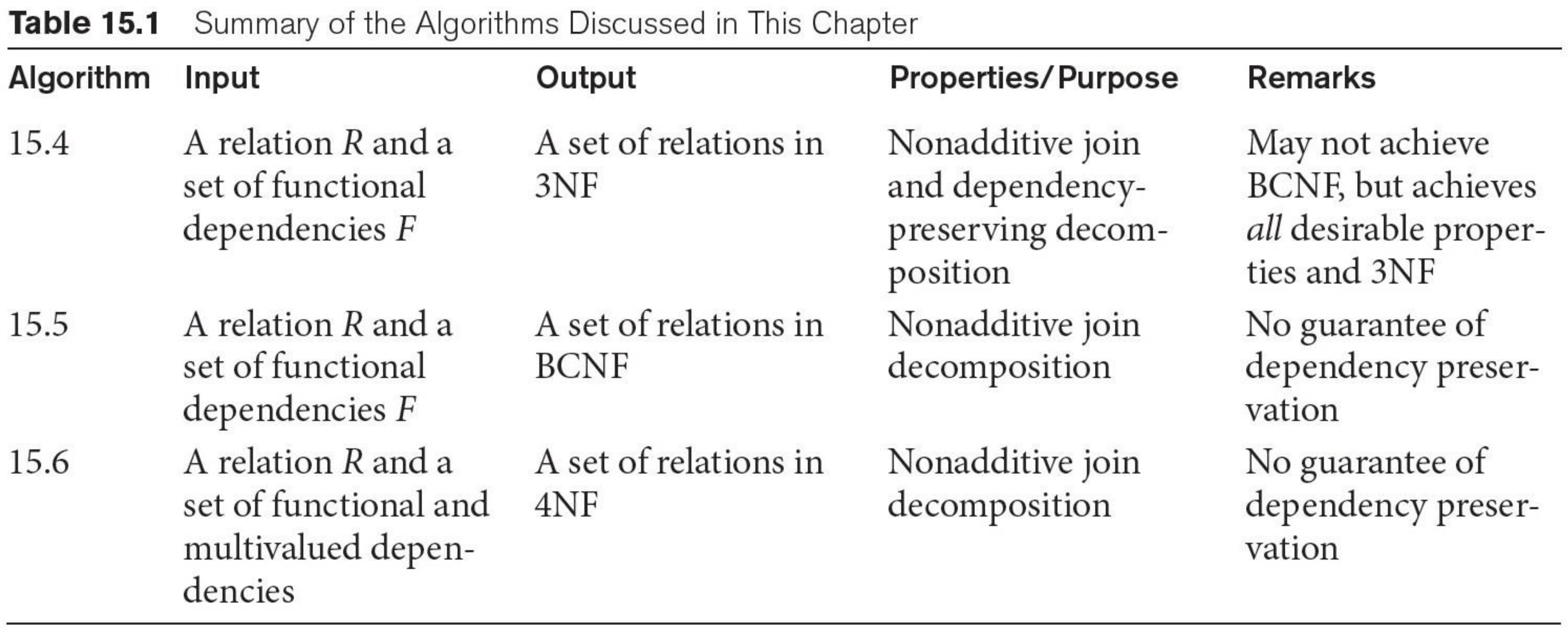


Table 15.1 Summary of the Algorithms Discussed in This Chapter

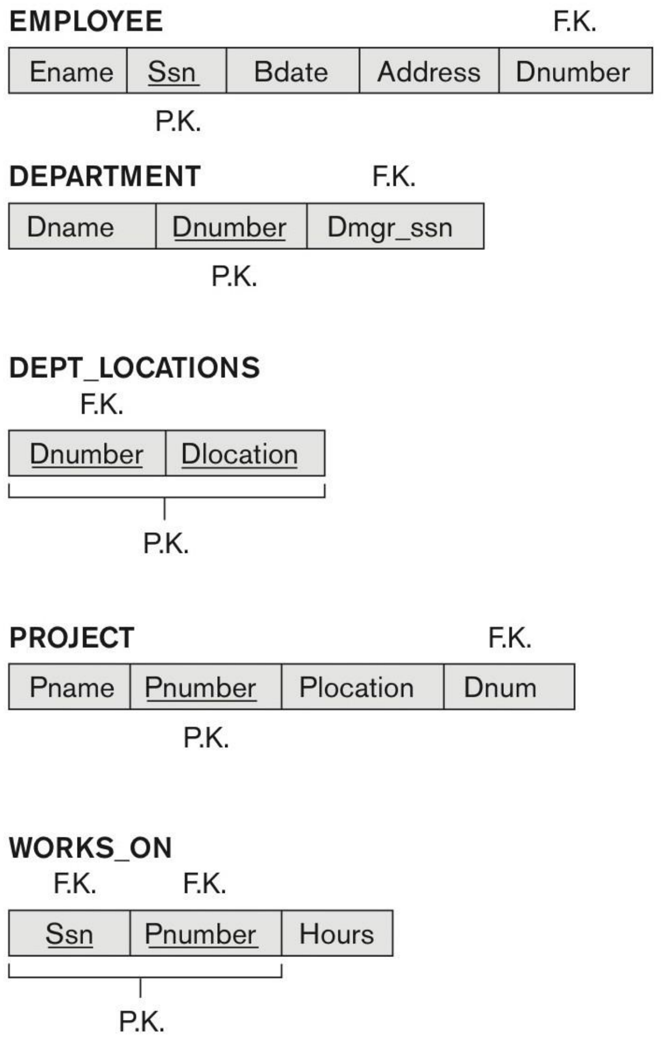
Algorithm	Input	Output	Properties/Purpose	Remarks
15.4	A relation *R* and a set of functional dependencies *F*	A set of relations in 3NF	Nonadditive join and dependency-preserving decomposition	May not achieve BCNF, but achieves *all* desirable properties and 3NF
15.5	A relation *R* and a set of functional dependencies *F*	A set of relations in BCNF	Nonadditive join decomposition	No guarantee of dependency preservation
15.6	A relation *R* and a set of functional and multivalued dependencies	A set of relations in 4NF	Nonadditive join decomposition	No guarantee of dependency preservation

- **Ignore algorithm 15.6 that does fourth normalization**

Some Other Dependencies in Relational Databases

- Besides FDs, MVDs (multivalued dependencies) and JDs (join dependencies) used for defining normal forms 1NF upto 5NF, some other dependencies have been proposed.
- We briefly cover Inclusion Dependencies and Arithmetic Dependencies
- Objective of Inclusion Dependencies:
  - To formalize two types of interrelational constraints which cannot be expressed using F.D.s or MVDs:
    - Referential integrity constraints
    - Class/subclass relationships

Figure 14.1 A simplified COMPANY relational database schema



Inclusion Dependencies (1)

Definition:

- An **inclusion dependency** $R.X < S.Y$ between two sets of attributes : X of relation schema R, and Y of relation schema S, specifies the constraint that, at any specific time when r is a relation state of R and s a relation state of S, we must have

$$\pi_X(r(R)) \subseteq \pi_Y(s(S))$$

- **Note**:
  - The $\subseteq$ (subset) relationship does not necessarily have to be a proper subset.
  - The sets of attributes on which the inclusion dependency is specified—X of R and Y of S—must have the same number of attributes.
  - In addition, the domains for each pair of corresponding attributes should be compatible.

Inclusion Dependencies (2)

- For example, we can specify the following inclusion dependencies on the relational schema in Figure 14.1:

DEPARTMENT.Dmgr_ssn < EMPLOYEE.Ssn

WORKS_ON.Ssn < EMPLOYEE.Ssn

EMPLOYEE.Dnumber < DEPARTMENT.Dnumber

PROJECT.Dnum < DEPARTMENT.Dnumber

WORKS_ON.Pnumber < PROJECT.Pnumber

DEPT_LOCATIONS.Dnumber < DEPARTMENT.Dnumber

Example of overlapping total Specialization

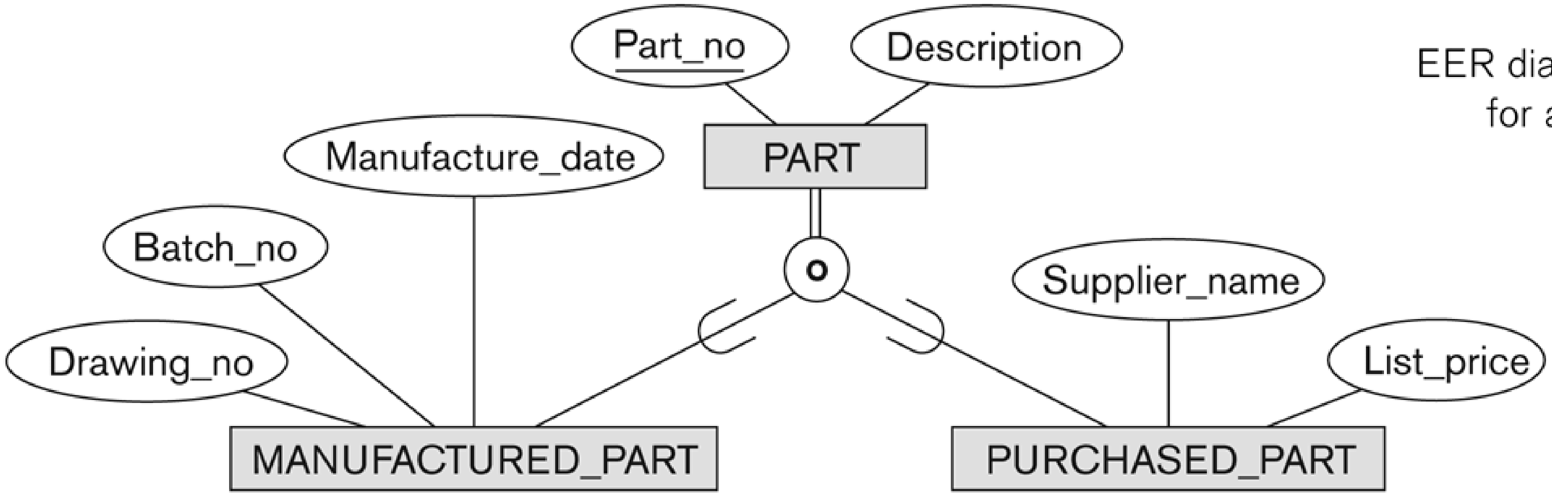


Figure 4.5
EER diagram notation for an overlapping (nondisjoint) specialization.

Inclusion Dependencies (3)

- **Suppose we map the above EER specialization (Fig. 4.5) into three tables:**

PART (Part#, Description)

MANUFACTURED_PARTS (Part#, Batch#, Drawing#, Manuf_date)

PURCHASED_PARTS (Part#, Supplier_name, List_price)

- **Then we would have the inclusion dependencies:**

MANUFACTURED_PARTS. Part# < PART. Part#

PURCHASED _PARTS. Part# < PART. Part#

< means subset

Functional Dependencies based on Arithmetic functions and procedures (1)

Arithmetic Functions:

- As long as a unique value of *Y* is associated with every *X*, we can still consider that the FD $X \rightarrow Y$ exists.

For example,consider the relation:

ORDER_LINE (Order#, Item#, Quantity, Unit_price, Extended_price, Discounted_price)

- each tuple represents an item from an order with a particular quantity, and the price per unit for that item. In this relation,

  (Quantity, Unit_price) $\rightarrow$ Extended_price by the formula

  Extended_price = Quantity * Unit_price .

- Hence, there is a unique value for Extended_price for every pair (Quantity, Unit_price), and thus it conforms to the definition of functional dependency.

Functional Dependencies based on Arithmetic functions and procedures (2)

Procedures:

- There may be a procedure that takes into account the quantity discounts, the type of item, and so on and computes a discounted price for the total quantity ordered for that item. Therefore, we can say
- (Item#, Quantity, Unit_price) $\rightarrow$ Discounted_price, or
- (Item#, Quantity, Extended_price) $\rightarrow$ Discounted_price.
- Here, the RHS value is a function of LHS parameters to be computed possibly by a complex procedure called COMPUTE_TOTAL_PRICE which may have to take into account various conditions and criteria
- The above dependencies are relevant during insertion/loading of data or query processing, but NOT relevant to normalization of the relation.

Recap of Lecture #12

- Functional Dependencies Revisited
- Designing a Set of Relations by Synthesis
- Properties of Relational Decompositions
- Algorithms for Relational Database Schema Design in 3NF and BCNF
- Nulls and Dangling Tuples
- Other Dependencies