MODULE – 4

NORMALIZATION: DATABASE DESIGN THEORY

INTRODUCTION TO NORMALIZATION USING FUNCTIONAL AND MULTIVALUED DEPENDENCIES

NORMALIZATION ALGORITHMS

Mr. C. R. Belavi, Dept. of CSE, HIT, NDS

INTRODUCTION TO NORMALIZATION USING FUNCTIONAL AND MULTIVALUED DEPENDENCIES

- Informal Design Guidelines for Relational Databases
- Functional Dependencies
- Normal Forms:
  - 1NF, 2NF, 3NF, BCNF, 4NF, 5NF
- Inference Rules
- Properties of Decompositions
- Algorithms for Relational Database Schema Design

Informal Design Guidelines for Relational Databases (1)

- What is relational database design?

  The grouping of attributes to form "good" relation schemas
- Two levels of relation schemas
  - The logical "user view" level
  - The storage "base relation" level
- Design is concerned mainly with base relations
- What are the criteria for "good" base relations?

Informal Design Guidelines for Relational Databases (2)

- We first discuss informal guidelines for good relational design
- Then we discuss formal concepts of functional dependencies and normal forms

  - 1NF (First Normal Form)

  - 2NF (Second Normal Form)

  - 3NF (Third Normal Form)

  - BCNF (Boyce-Codd Normal Form)

Semantics of the Relation Attributes

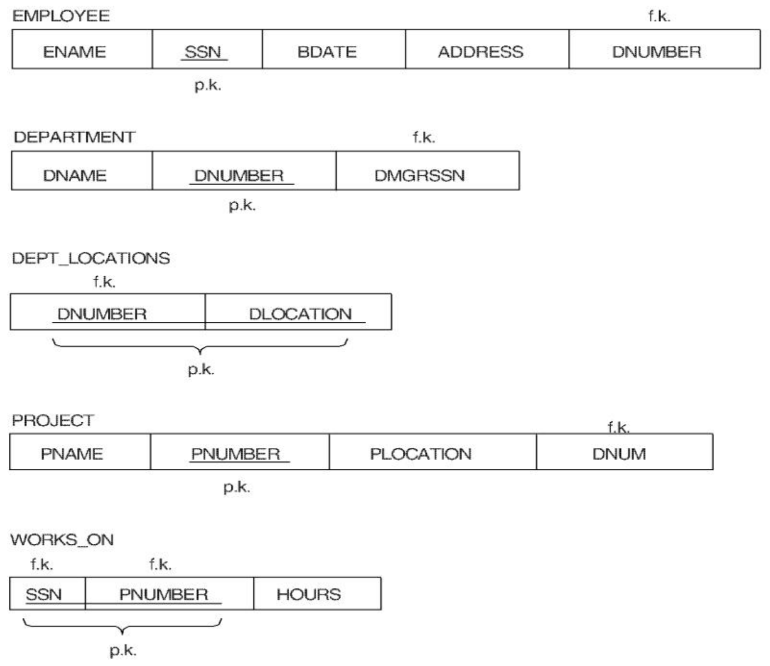
GUIDELINE 1: Informally, each tuple in a relation should represent one entity or relationship instance. (Applies to individual relations and their attributes).

- Attributes of different entities (EMPLOYEEs, DEPARTMENTs, PROJECTs) should not be mixed in the same relation
- Only foreign keys should be used to refer to other entities
- Entity and relationship attributes should be kept apart as much as possible.

Bottom Line: Design a schema that can be explained easily relation by relation. The semantics of attributes should be easy to interpret.

A simplified COMPANY relational database schema

Figure 14.1 Simplified version of the COMPANY relational database schema.



Note: The above figure is now called Figure 10.1 in Edition 4

Redundant Information in Tuples and Update Anomalies

- Mixing attributes of multiple entities may cause problems
- Information is stored redundantly wasting storage
- Problems with update anomalies
  - Insertion anomalies
  - Deletion anomalies
  - Modification anomalies

EXAMPLE OF AN UPDATE ANOMALY (1)

Consider the relation:

EMP_PROJ (<u>Emp#, Proj#,</u> Ename, Pname, No_hours)

- **Update Anomaly:** Changing the name of project number P1 from “Billing” to “Customer-Accounting” may cause this update to be made for all 100 employees working on project P1.

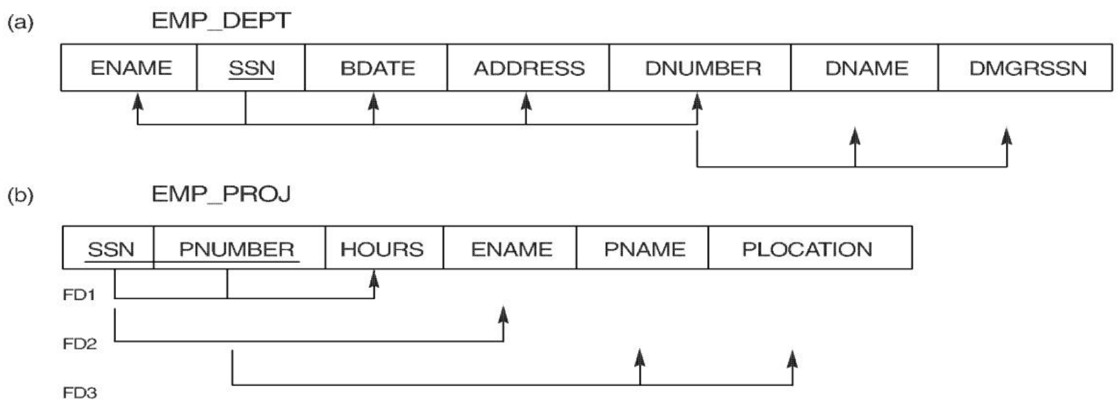
EXAMPLE OF AN UPDATE ANOMALY (2)

- **Insert Anomaly:** Cannot insert a project unless an employee is assigned to .

  Inversely - Cannot insert an employee unless an he/she is assigned to a project.
- **Delete Anomaly:** When a project is deleted, it will result in deleting all the employees who work on that project. Alternately, if an employee is the sole employee on a project, deleting that employee would result in deleting the corresponding project.

Two relation schemas suffering from update anomalies

Figure 14.3 Two relation schemas and their functional dependencies. Both suffer from update anomalies. (a) The EMP_DEPT relation schema. (b) The EMP_PROJ relation schema.



Example States for EMP_DEPT and EMP_PROJ

Figure 14.4 Example relations for the schemas in Figure 14.3 that result from applying NATURAL JOIN to the relations in Figure 14.2. These may be stored as base relations for performance reasons.

EMP_DEPT

ENAME	SSN	BDATE	ADDRESS	DNUMBER	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 FireOak,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

EMP_PROJ

SSN	PNUMBER	HOURS	ENAME	PNAME	PLOCATION
123456789	1	32.5	Smith,John B.	ProductX	Bellaire
123456789	2	7.5	Smith,John B.	ProductY	Sugarland
666884444	3	40.0	Narayan,Ramesh K.	ProductZ	Houston
453453453	1	20.0	English,Joyce A.	ProductX	Bellaire
453453453	2	20.0	English,Joyce A.	ProductY	Sugarland
333445555	2	10.0	Wong,Franklin T.	ProductY	Sugarland
333445555	3	10.0	Wong,Franklin T.	ProductZ	Houston
333445555	10	10.0	Wong,Franklin T.	Computerization	Stafford
333445555	20	10.0	Wong,Franklin T.	Reorganization	Houston
999887777	30	30.0	Zelaya,Alicia J.	Newbenefits	Stafford
999887777	10	10.0	Zelaya,Alicia J.	Computerization	Stafford
987987987	10	35.0	Jabbar,Ahmad V.	Computerization	Stafford
987987987	30	5.0	Jabbar,Ahmad V.	Newbenefits	Stafford
987654321	30	20.0	Wallace,Jennifer S.	Newbenefits	Stafford
987654321	20	15.0	Wallace,Jennifer S.	Reorganization	Houston
888665555	20	null	Borg,James E.	Reorganization	Houston

Guideline to Redundant Information in Tuples and Update Anomalies

- **GUIDELINE 2:** Design a schema that does not suffer from the insertion, deletion and update anomalies. If there are any present, then note them so that applications can be made to take them into account

Null Values in Tuples

GUIDELINE 3: Relations should be designed such that their tuples will have as few NULL values as possible

- Attributes that are NULL frequently could be placed in separate relations (with the primary key)
- Reasons for nulls:
  - attribute not applicable or invalid
  - attribute value unknown (may exist)
  - value known to exist, but unavailable

Spurious Tuples

- Bad designs for a relational database may result in erroneous results for certain JOIN operations
- The "lossless join" property is used to guarantee meaningful results for join operations

GUIDELINE 4: The relations should be designed to satisfy the lossless join condition. No spurious tuples should be generated by doing a natural-join of any relations.

Spurious Tuples (2)

There are two important properties of decompositions:

(a) non-additive or losslessness of the corresponding join

(b) preservation of the functional dependencies.

Note that property (a) is extremely important and *cannot* be sacrificed. Property (b) is less stringent and may be sacrificed.

Functional Dependencies (1)

- Functional dependencies (FDs) are used to specify *formal measures* of the "goodness" of relational designs
- FDs and keys are used to define **normal forms** for relations
- FDs are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes
- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y

Functional Dependencies (2)

- X -> Y holds if whenever two tuples have the same value for X, they *must have* the same value for Y
- For any two tuples t1 and t2 in any relation instance r(R): *If* t1[X]=t2[X], *then* t1[Y]=t2[Y]
- X -> Y in R specifies a *constraint* on all relation instances r(R)
- Written as X -> Y; can be displayed graphically on a relation schema as in Figures. (denoted by the arrow:).
- FDs are derived from the real-world constraints on the attributes

Examples of FD constraints (1)

- social security number determines employee name

  SSN -> ENAME
- project number determines project name and location

  PNUMBER -> {PNAME, PLOCATION}
- employee ssn and project number determines the hours per week that the employee works on the project

  {SSN, PNUMBER} -> HOURS

Examples of FD constraints (2)

- An FD is a property of the attributes in the schema R
- The constraint must hold on *every relation instance* r(R)
- If K is a key of R, then K functionally determines all attributes in R (since we never have two distinct tuples with t1[K]=t2[K])

Normal Forms Based on Primary Keys

- Normalization of Relations
- Practical Use of Normal Forms
- Definitions of Keys and Attributes Participating in Keys
- First Normal Form
- Second Normal Form
- Third Normal Form

Normalization of Relations (1)

- **Normalization:** The process of decomposing unsatisfactory "bad" relations by breaking up their attributes into smaller relations

- **Normal form:** Condition using keys and FDs of a relation to certify whether a relation schema is in a particular normal form

Normalization of Relations (2)

- 2NF, 3NF, BCNF based on keys and FDs of a relation schema
- 4NF based on keys, multi-valued dependencies : MVDs; 5NF based on keys, join dependencies : JDs
- Additional properties may be needed to ensure a good relational design (lossless join, dependency preservation)

- **Lossless join or nonadditive join property**
  - Guarantees that the spurious tuples will not be generated
- **The dependency preservation property**
  - Ensures that each functional dependency is represented in some individual relation resulting after decomposition

Practical Use of Normal Forms

- **Normalization** is carried out in practice so that the resulting designs are of high quality and meet the desirable properties
- The practical utility of these normal forms becomes questionable when the constraints on which they are based are **hard to understand** or to **detect**
- The database designers ***need not*** normalize to the highest possible normal form. (usually up to 3NF, BCNF or 4NF)
- **Denormalization:** the process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

Definitions of Keys and Attributes Participating in Keys (1)

- A **superkey** of a relation schema $R = \{A_1, A_2, ..., A_n\}$ is a set of attributes S <u>*subset-of*</u> R with the property that no two tuples t_1 and t_2 in any legal relation state r of R will have $t_1[S] = t_2[S]$

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

Definitions of Keys and Attributes Participating in Keys (2)

- If a relation schema has more than one key, each is called a **candidate key.** One of the candidate keys is *arbitrarily* designated to be the **primary key,** and the others are called *secondary keys.*
- A **Prime attribute** must be a member of *some candidate key*
- A **Nonprime attribute** is not a prime attribute—that is, it is not a member of any candidate key.

First Normal Form

- Disallows composite attributes, multivalued attributes, and **nested relations;** attributes whose values *for an individual tuple* are non-atomic

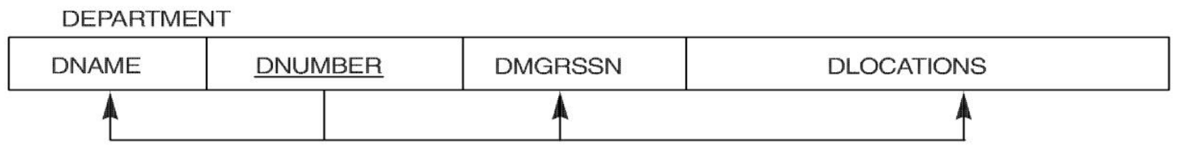
- Considered to be part of the definition of relation

- 1NF Definition:
  - **It states that the domain of an attribute must include only atomic (simple) values and that the value of any attribute in a tuple must be a single value from the domain of that attribute**

Normalization into 1NF

Figure 14.8 Normalization into 1NF. (a) Relation schema that is not in 1NF. (b) Example relation instance. (c) 1NF relation with redundancy.

(a)



(b)

DEPARTMENT

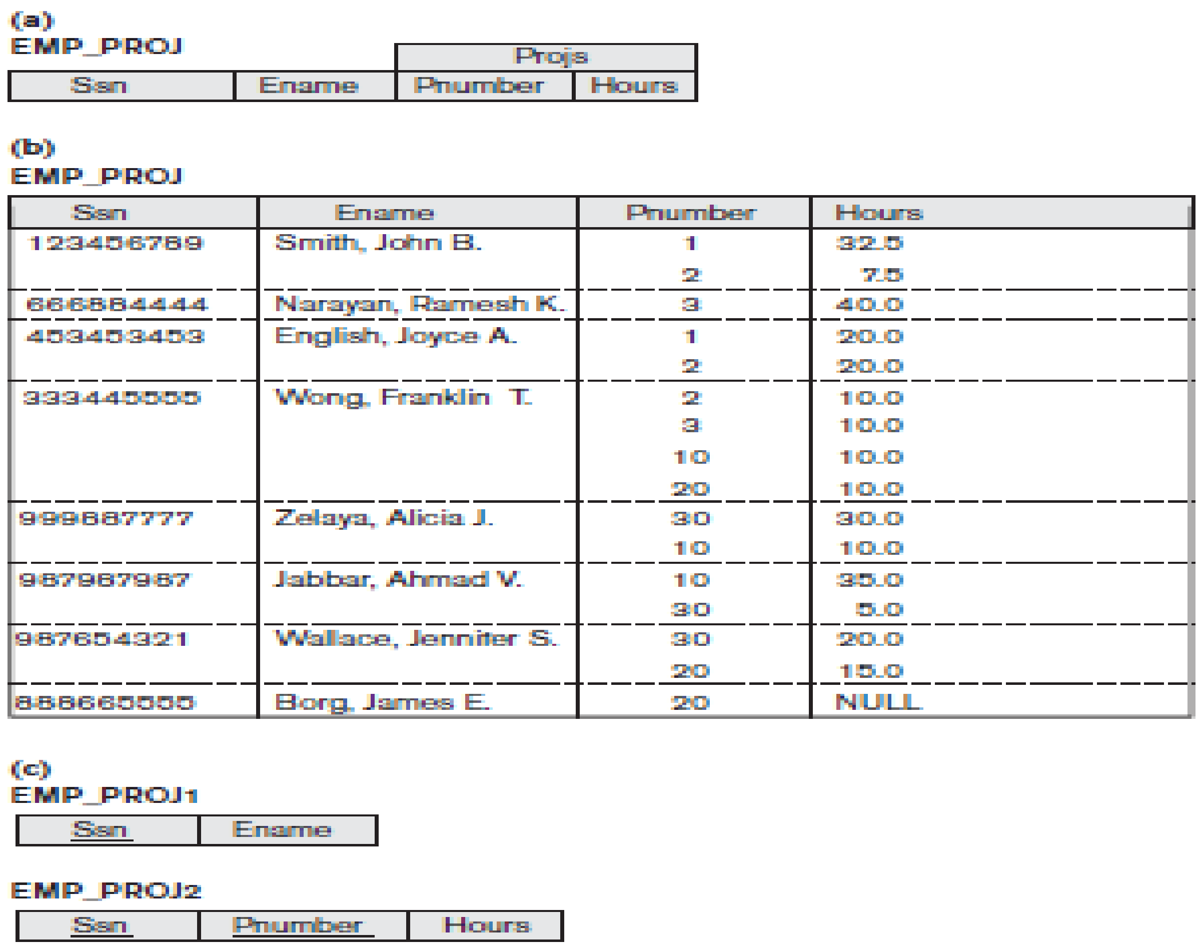
DNAME	DNUMBER	DMGRSSN	DLOCATIONS
Research	5	333445555	{Bellaire, Sugarland, Houston}
Administration	4	987654321	{Stafford}
Headquarters	1	888665555	{Houston}

(c)

DEPARTMENT

DNAME	DNUMBER	DMGRSSN	DLOCATION
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Normalization nested relations into 1NF



(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

(c)
EMP_PROJ1

Ssn	Ename

EMP_PROJ2

Ssn	Pnumber	Hours

Figure 15.10
Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Sample extension of the EMP_PROJ relation showing nested relations within each tuple. (c) Decomposition of EMP_PROJ into relations EMP_PROJ1 and EMP_PROJ2 by propagating the primary key.

Second Normal Form (1)

- Uses the concepts of **FDs**, **primary key**

Definitions:

- **Prime attribute** - attribute that is member of the primary key K
- **Full functional dependency** - a FD Y -> Z where removal of any attribute from Y means the FD does not hold any more

Examples: - {SSN, PNUMBER} -> HOURS is a full FD since neither SSN -> HOURS nor PNUMBER -> HOURS hold

- {SSN, PNUMBER} -> ENAME is *not* a full FD (it is called a *partial dependency*) since SSN -> ENAME also holds

Second Normal Form (2)

- 2NF Definition
  - **A relation schema R is in second normal form (2NF) if every non-prime attribute A in R is fully functionally dependent on the primary key of R**

- R can be decomposed into 2NF relations via the process of 2NF normalization

Normalizing into 2NF and 3NF

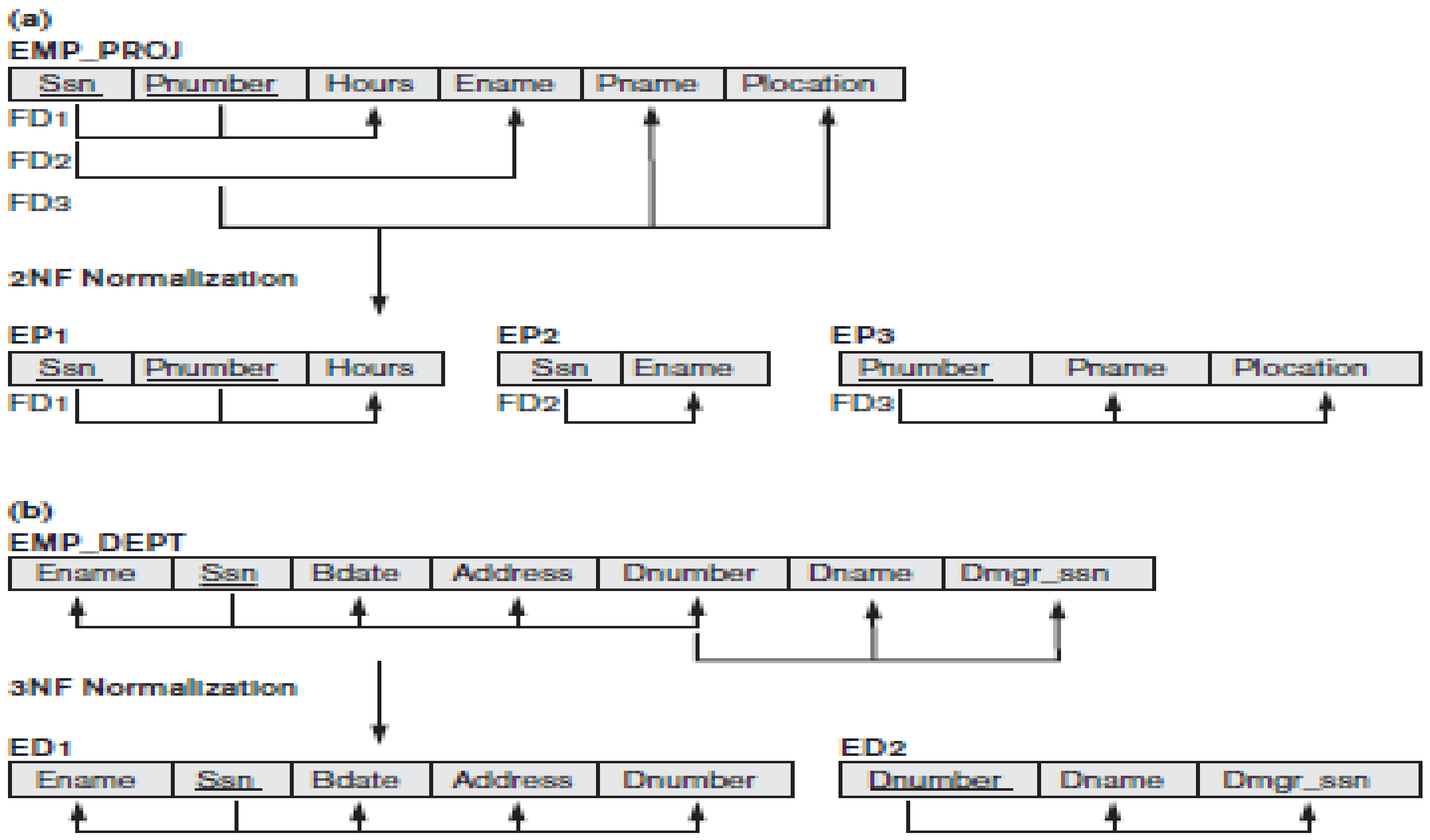
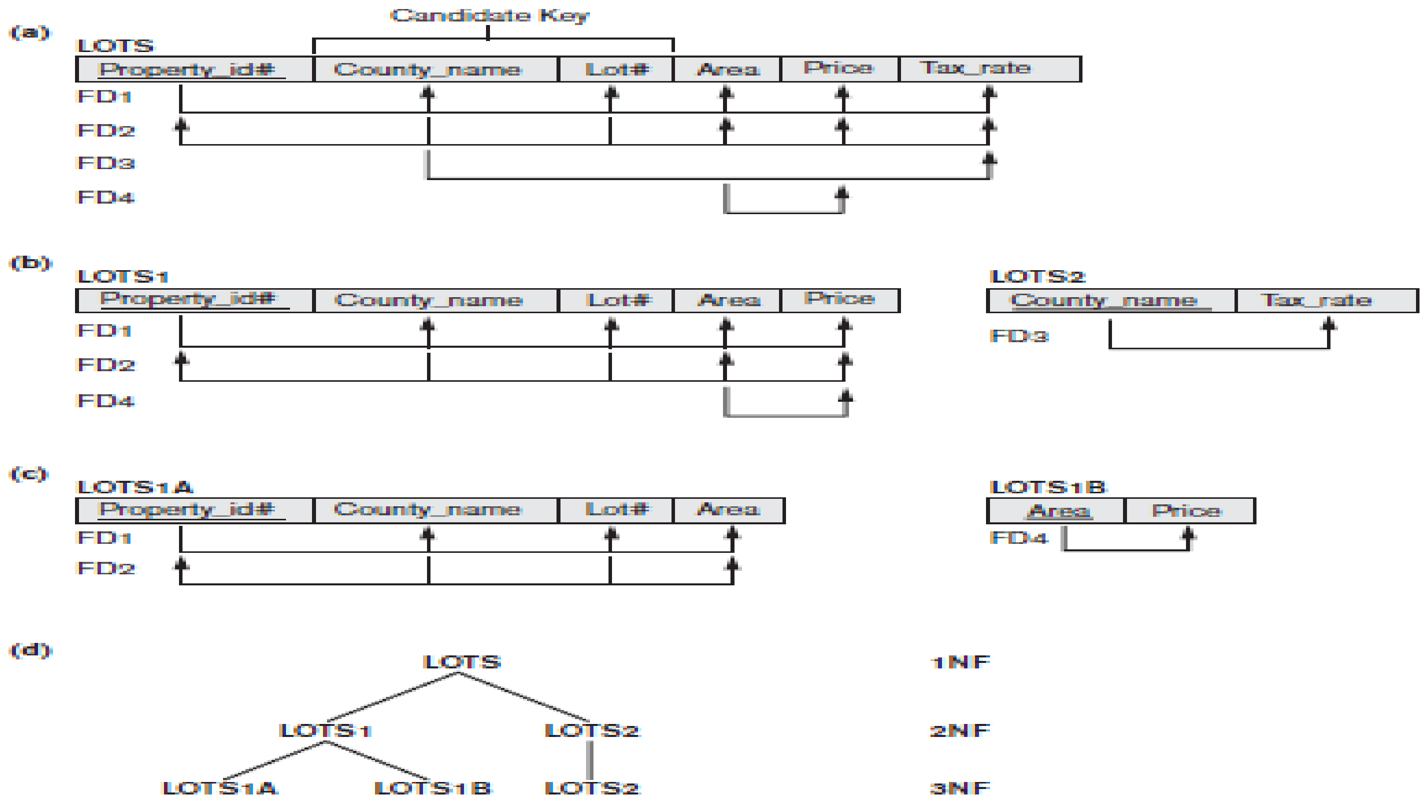


Figure 15.11
Normalizing into 2NF and 3NF. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

Normalization into 2NF and 3NF

Figure 15.12
Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Summary of the progressive normalization of LOTS.



Third Normal Form (1)

Definition:

- **Transitive functional dependency** - a FD X -> Z that can be derived from two FDs X -> Y and Y -> Z

  Examples:

  - SSN -> DMGRSSN is a *transitive* FD since

  SSN -> DNUMBER and DNUMBER -> DMGRSSN hold

  - SSN -> ENAME is *non-transitive* since there is no set of attributes X where SSN -> X and X -> ENAME

Third Normal Form (2)

- 3NF Definition
  - **A relation schema R is in third normal form (3NF) if it satisfies 2NF *and* no non-prime attribute of R is transitively dependent on the primary key**

- R can be decomposed into 3NF relations via the process of 3NF normalization

NOTE:

In X -> Y and Y -> Z, with X as the primary key, we consider this a problem only if Y is <u>not</u> a candidate key. When Y is a candidate key, there is no problem with the transitive dependency .

E.g., Consider EMP (SSN, Emp#, Salary).

Here, SSN -> Emp# -> Salary and Emp# is a candidate key.

General Normal Form Definitions (For Multiple Keys) (1)

- The above definitions consider the primary key only
- The following more general definitions take into account relations with multiple candidate keys
- A relation schema R is in **second normal form** (**2NF**) if every non-prime attribute A in R is fully functionally dependent on *every key* of R

General Normal Form Definitions (2)

Definition:

- **Superkey** of relation schema R - a set of attributes S of R that contains a key of R
- A relation schema R is in **third normal form** (**3NF**) if whenever a FD X -> A holds in R, then either:

    (a) X is a superkey of R, or

    (b) A is a prime attribute of R

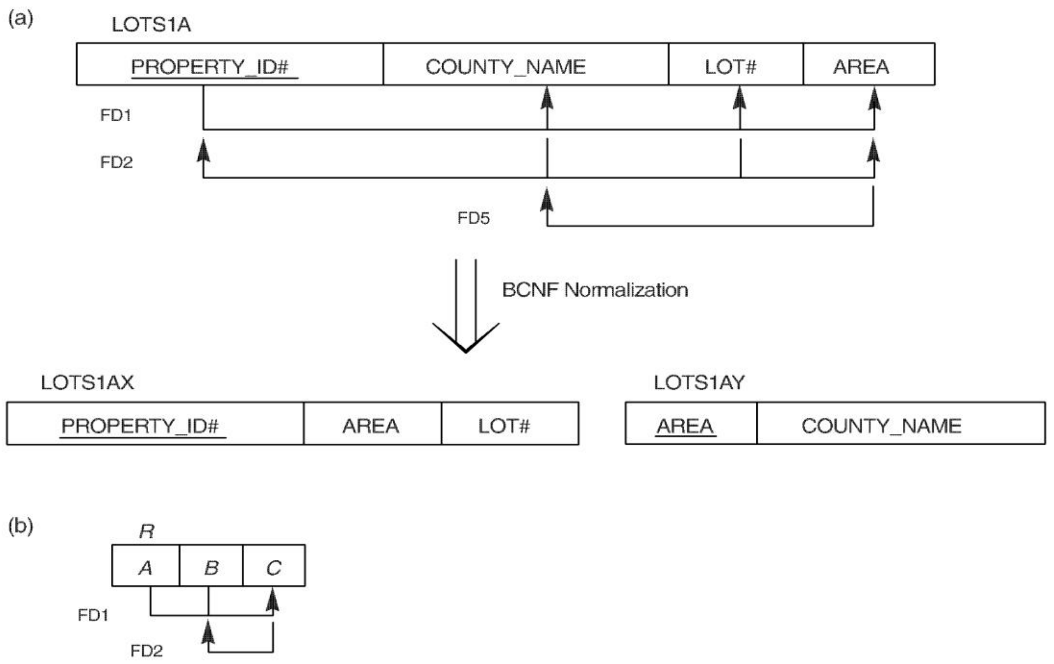
NOTE: Boyce-Codd normal form disallows condition (b) above

BCNF (Boyce-Codd Normal Form)

- A relation schema R is in **Boyce-Codd Normal Form** (**BCNF**) if whenever an FD X -> A holds in R, then X is a superkey of R
- Each normal form is strictly stronger than the previous one
  - Every 2NF relation is in 1NF
  - Every 3NF relation is in 2NF
  - Every BCNF relation is in 3NF
- There exist relations that are in 3NF but not in BCNF
- The goal is to have each relation in BCNF (or 3NF)

Boyce-Codd normal form

Figure 14.12 Boyce-Codd normal form. (a) BCNF normalization with the dependency of FD2 being "lost" in the decomposition. (b) A relation R in 3NF but not in BCNF.



A relation TEACH that is in 3NF but not in BCNF

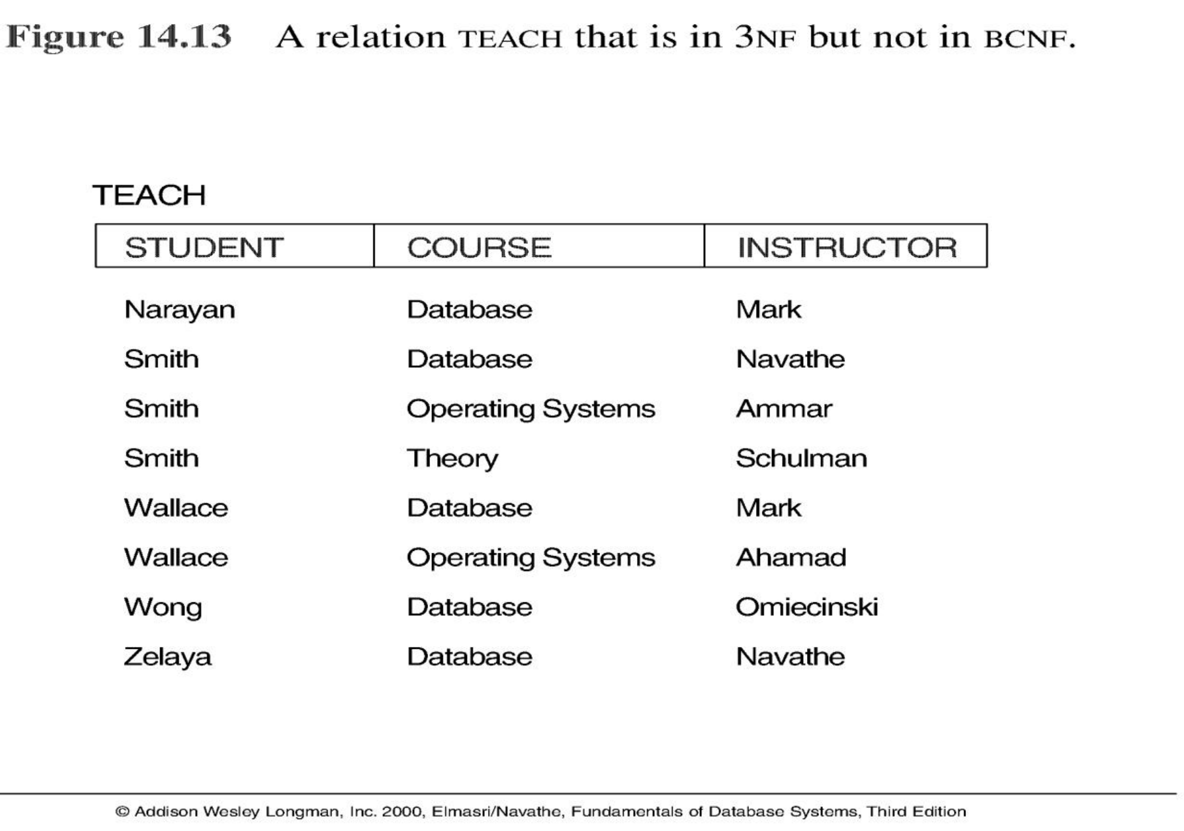


Figure 14.13 A relation TEACH that is in 3NF but not in BCNF.

TEACH

STUDENT	COURSE	INSTRUCTOR
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe

Achieving the BCNF by Decomposition (1)

- Two FDs exist in the relation TEACH:

  fd1: { student, course} -> instructor

  fd2: instructor -> course
- {student, course} is a candidate key for this relation and that the dependencies shown follow the pattern in Figure 10.12 (b). So this relation is in 3NF <u>but not in</u> BCNF
- A relation **NOT** in BCNF should be decomposed so as to meet this property, while possibly forgoing the preservation of all functional dependencies in the decomposed relations. (See Algorithm 11.3)

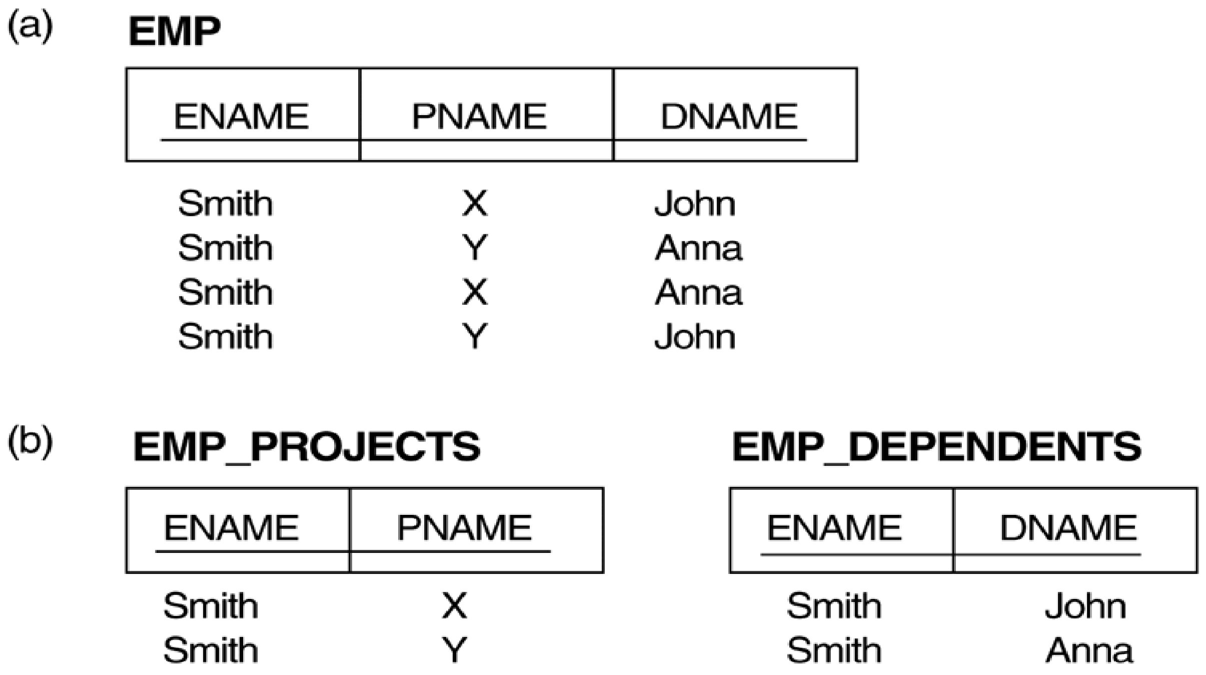
Achieving the BCNF by Decomposition (2)

- Three possible decompositions for relation TEACH
  1. {student, instructor} and {student, course}
  2. {course, instructor } and {course, student}
  3. {instructor, course } and {instructor, student}
- All three decompositions will lose fd1. We have to settle for sacrificing the functional dependency preservation. But we cannot sacrifice the non-additivity property after decomposition.
- Out of the above three, only the 3rd decomposition will not generate spurious tuples after join.(and hence has the non-additivity property).
- A test to determine whether a binary decomposition (decomposition into two relations) is nonadditive (lossless) is discussed in section 11.1.4 under Property LJ1. Verify that the third decomposition above meets the property.

Multivalued Dependencies and Fourth Normal Form (1)

(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

Multivalued Dependencies and Fourth Normal Form (1)

(c) The relation SUPPLY with no MVDs is in 4NF but not in 5NF if it has the JD(R1, R2, R3). (d) Decomposing the relation SUPPLY into the 5NF relations R1, R2, and R3.

(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

(d) **R1**

SNAME	PARTNAME
Smith	Bolt
Smith	Nut
Adamsky	Bolt
Walton	Nut
Adamsky	Nail

R2

SNAME	PROJNAME
Smith	ProjX
Smith	ProjY
Adamsky	ProjY
Walton	ProjZ
Adamsky	ProjX

R3

PARTNAME	PROJNAME
Bolt	ProjX
Nut	ProjY
Bolt	ProjY
Nut	ProjZ
Nail	ProjX

Multivalued Dependencies and Fourth Normal Form (2)

Definition:

- A **multivalued dependency** (**MVD**) $X \longrightarrow\!\!> Y$ 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, where we use Z to denote $(R - (X \cup Y))$:
  - $t_3[X] = t_4[X] = t_1[X] = t_2[X]$.
  - $t_3[Y] = t_1[Y]$ and $t_4[Y] = t_2[Y]$.
  - $t_3[Z] = t_2[Z]$ and $t_4[Z] = t_1[Z]$.
- An MVD $X \longrightarrow\!\!> Y$ in R is called a **trivial MVD** if (a) Y is a subset of X, or (b) $X \cup Y = R$.

Multivalued Dependencies and Fourth Normal Form (3)

- **Inference Rules for Functional and Multivalued 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**): $\{X \twoheadrightarrow Y, Y \twoheadrightarrow Z\} \models X \twoheadrightarrow (Z$ 2 $Y)$.
  - 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 $X \rightarrow Z$.

Multivalued Dependencies and Fourth Normal Form (4)

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*.

Multivalued Dependencies and Fourth Normal Form (5)

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

Multivalued Dependencies and Fourth Normal Form (6)

Lossless (Non-additive) 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
    - $(R_1 \cap R_2) \twoheadrightarrow (R_2 - R_1)$).

Multivalued Dependencies and Fourth Normal Form (7)

Algorithm 11.5: 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 $(X \cup Y)$;
   };
```

Join Dependencies and Fifth Normal Form (1)

Definition:

- A **join dependency** (**JD**), denoted by JD(R_1, R_2, ..., R_n), 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 non-additive join decomposition into R_1, R_2, ..., R_n; that is, for every such r we have
  - $$* (\pi_{R1}(r), \pi_{R2}(r), ..., \pi_{Rn}(r)) = r$$

  ***Note**: an MVD is a special case of a JD where n = 2.*
- A join dependency JD(R_1, R_2, ..., R_n), specified on relation schema R, is a **trivial JD** if one of the relation schemas R_i in JD(R_1, R_2, ..., R_n) is equal to R.

Join Dependencies and Fifth Normal Form (2)

<u>Definition:</u>

- 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(R_1, R_2, ..., R_n) in F^+ (that is, implied by *F*),
    - every R_i is a superkey of *R*.

Relation SUPPLY with Join Dependency and conversion to Fifth Normal Form

Figure 11.4
Fourth and fifth normal forms.
(a) The EMP relation with two MVDs: Ename $\twoheadrightarrow$ Pname and Ename $\twoheadrightarrow$ Dname.
(b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS.
(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

NORMALIZATION ALGORITHMS

Closure of f

- The set of all dependencies that include F as well as all dependencies that can be inferred from F is called the closure of F; it is denoted by F+.

- Example…

$F = \{Ssn \rightarrow \{Ename, Bdate, Address, Dnumber\}, Dnumber \rightarrow \{Dname, Dmgr\_ssn\}\}$

- Some of the additional functional dependencies that we can infer from F are the following:

Ssn → {Dname, Dmgr_ssn}

Ssn → Ssn

Dnumber → Dname

2.2 Inference Rules for FDs (1)

- Given a set of FDs F, we can *infer* additional FDs that hold whenever the FDs in F hold

Armstrong's inference rules:

IR1. (**Reflexive**) If Y *subset-of* X, then X -> Y

IR2. (**Augmentation**) If X -> Y, then XZ -> YZ

(Notation: XZ stands for X U Z)

IR3. (**Transitive**) If X -> Y and Y -> Z, then X -> Z

- IR1, IR2, IR3 form a *sound* and *complete* set of inference rules

Inference Rules for FDs (2)

Some **additional inference rules** that are useful:

(**Decomposition**) If X -> YZ, then X -> Y and X -> Z

(**Union**) If X -> Y and X -> Z, then X -> YZ

(**Psuedotransitivity**) If X -> Y and WY -> Z, then WX -> Z

- The last three inference rules, as well as any other inference rules, can be deduced from IR1, IR2, and IR3 (completeness property)

PROOF

- Proof of IR1.
  - Suppose that $X \supseteq Y$ and that two tuples t1 and t2 exist in some relation instance r of R such that t1 [X] = t2 [X]. Then t1[Y] = t2[Y] because $X \supseteq Y$; hence, $X \rightarrow Y$ must hold in r.
- Proof of IR2 (by contradiction).
  - Assume that $X \rightarrow Y$ holds in a relation instance r of R but that $XZ \rightarrow YZ$ does not hold. Then there must exist two tuples t1 and t2 in r such that (1) t1 [X] = t2 [X], (2) t1 [Y] = t2 [Y], (3) t1 [XZ] = t2 [XZ], and (4) t1 [YZ] $\neq$ t2 [YZ]. This is not possible because from (1) and (3) we deduce (5) t1 [Z] = t2 [Z], and from (2) and (5) we deduce (6) t1 [YZ] = t2 [YZ], contradicting (4).
- Proof of IR3.
  - Assume that (1) $X \rightarrow Y$ and (2) $Y \rightarrow Z$ both hold in a relation r. Then for any two tuples t1 and t2 in r such that t1 [X] = t2 [X], we must have (3) t1 [Y] = t2 [Y], from assumption (1); hence we must also have (4) t1 [Z] = t2 [Z] from (3) and assumption (2); thus $X \rightarrow Z$ must hold in r.

CONTD

- Proof of IR4 (Using IR1 through IR3).
  - $X \longrightarrow YZ$ (given).
  - $YZ \longrightarrow Y$ (using IR1 and knowing that $YZ \supseteq Y$).
  - $X \longrightarrow Y$ (using IR3 on 1 and 2).
- Proof of IR5 (using IR1 through IR3).
  - $X \longrightarrow Y$ (given).
  - $X \longrightarrow Z$ (given).
  - $X \longrightarrow XY$ (using IR2 on 1 by augmenting with X; notice that $XX = X$).
  - $XY \longrightarrow YZ$ (using IR2 on 2 by augmenting with Y).
  - $X \longrightarrow YZ$ (using IR3 on 3 and 4).
- Proof of IR6 (using IR1 through IR3).
  - $X \longrightarrow Y$ (given).
  - $WY \longrightarrow Z$ (given).
  - $WX \longrightarrow WY$ (using IR2 on 1 by augmenting with W).
  - $WX \longrightarrow Z$ (using IR3 on 3 and 2).

- The inference rules IR1 through IR3 are **sound** and **complete**
  - **sound** because given a set of functional dependencies F specified on a relation schema R, any dependency that we can infer from F by using IR1 through IR3 holds in every relation state r or R that satisfies the dependencies in F.

  - **complete** because using IR1 through IR3 repeatedly to infer dependencies until no more dependencies can be inferred results in the complete set of all possible dependencies that can be inferred from F.

Closure of x under f

- For each set of attributes *X*, *we determine the set* X^+ of *attributes* that are functionally determined by *X based on F;* X^+ *is called the* **closure of X under F.**

Algorithm: Determining X+, the Closure of X under F

Input: A set F of FDs on a relation schema R, and a set of attributes X, which is a subset of R.

```
$X^+ := X$;
repeat
    $oldX^+ := X^+$;
    for each functional dependency $Y \rightarrow Z$ in F do
        if $X^+ \supseteq Y$ then $X^+ := X^+ \cup Z$;
until ($X^+ = oldX^+$);
```

Equivalence of sets of functional dependencies

- A set of functional dependencies F is said to **cover** another set of functional dependencies E if every FD in E is also in F+; that is, if every dependency in E can be inferred from F; alternatively, we can say that E is **covered by F.**

- Two sets of functional dependencies E and F are **equivalent if** E+ = F+. Therefore, equivalence means that every FD in E can be inferred from F, and every FD in F can be inferred from E; that is, E is equivalent to F if both the conditions—E covers F and F covers E—hold.

Minimal Sets of FDs

- A set of functional dependencies F to be minimal if it satisfies the following conditions:
  1. Every dependency in F has a single attribute for its right-hand side.
  2. We cannot replace any dependency $X \rightarrow A$ in F with a dependency $Y \rightarrow A$, where Y is a proper subset of X, and still have a set of dependencies that is equivalent to F.
  3. We cannot remove any dependency from F and still have a set of dependencies that is equivalent to F.

Minimal cover

- A minimal cover of a set of functional dependencies E is a minimal set of dependencies (in the standard canonical form and without redundancy) that is equivalent to E

Algorithm: Finding a Minimal Cover F for a Set of Functional Dependencies E

Input: A set of functional dependencies E.

1. Set *F* := *E*.
2. Replace each functional dependency *X*⟶{*A*1, *A*2, ..., *An*} *in F by the n functional* dependencies *X*⟶*A*1, *X*⟶*A*2, ..., *X*⟶*An*.
3. For each functional dependency *X*⟶*A in F* for each attribute *B that is an element of X* if { {*F* – {*X*⟶*A*} } *U* { (*X* – {*B*}) ⟶*A*} } *is equivalent to F* then replace *X*⟶*A with* (*X* – {*B*}) ⟶*A in F*.
4. For each remaining functional dependency *X*⟶*A in F* if {*F* – {*X*⟶*A*} } *is equivalent to F*, then remove *X*⟶*A from F*.

Example

Find the minimal cover for the following set of functional dependencies:

$$E : \{B \longrightarrow A,\ D \longrightarrow A,\ AB \longrightarrow D\}.$$

SOLUTION

- All above dependencies are in canonical form (that is, they have only one attribute on the right-hand side), so we have completed step 1 of Algorithm and can proceed to step 2. In step 2 we need to determine if *AB→D has* any redundant attribute on the left-hand side; that is, can it be replaced by *B→D or A→D?*
- Since B →A, by augmenting with *B on both sides (IR2), we have BB → AB,* or *B→AB (i). However, AB→D as given (ii).*
- Hence by the transitive rule (IR3), we get from (i) and (ii), *B → D. Thus AB→D may be replaced by B→D.*
- We now have a set equivalent to original *E, say E: {B→A, D→A, B→D}*. No further reduction is possible in step 2 since all FDs have a single attribute on the left-hand side.
- In step 3 we look for a redundant FD in *E. By using the transitive rule on B → D and D → A, we derive B → A. Hence B → A is redundant in E and* can be eliminated.
- Therefore, the minimal cover of *E is {B→D, D→A}*.

DESIGNING A SET OF RELATIONS (1)

- **The Approach of Relational Synthesis (Bottom-up Design):**
  - 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 *set of relations* in a relational database are satisfactory (see Algorithms 11.2 and 11.4).

DESIGNING A SET OF RELATIONS (2)

- **Goals:**
  - Lossless join property (a must)
    - Algorithm 11.1 tests for general losslessness.
  - Dependency preservation property
    - Algorithm 11.3 decomposes a relation into BCNF components by sacrificing the dependency preservation.
  - Additional normal forms
    - 4NF (based on multi-valued dependencies)
    - 5NF (based on join dependencies)

1. Properties of Relational Decompositions (1)

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

Properties of Relational Decompositions (2)

- **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 Ri in the decomposition so that no attributes are "lost".

Properties of Relational Decompositions (2)

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

Properties of Relational Decompositions (3)

- **Dependency Preservation Property of a Decomposition:**
  - Definition: Given a set of dependencies F on R, the **projection** of F on R_i, denoted by $p_{Ri}(F)$ where R_i is a subset of R, is the set of dependencies $X \rightarrow Y$ in F^+ such that the attributes in $X \cup Y$ 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 (4)

- **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

      $$((\pi_{R1}(F)) \cup \ldots \cup (\pi_{Rm}(F)))^+ = F^+$$
    - (See examples in Fig 10.12a and Fig 10.11)
- Claim 1:
  - It is always possible to find a dependency-preserving decomposition D with respect to F such that each relation Ri in D is in 3nf.

Properties of Relational Decompositions (5)

- **Lossless (Non-additive) 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**"

Properties of Relational Decompositions (6)

- **Lossless (Non-additive) Join Property of a Decomposition (cont.):**
- **Algorithm 11.1: 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

Properties of Relational Decompositions (7)

- **Lossless (Non-additive) Join Property of a Decomposition (cont.):**
- **Algorithm 11.1: Testing for Lossless Join Property**

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.

Properties of Relational Decompositions (8)

Lossless (nonadditive) join test for *n*-ary decompositions.
(a) Case 1: Decomposition of EMP_PROJ into EMP_PROJ1 and EMP_LOCS fails test.
(b) A decomposition of EMP_PROJ that has the lossless join property.

(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→ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→HOURS}

$$\begin{array}{c|cccccc}
 & \text{SSN} & \text{ENAME} & \text{PNUMBER} & \text{PNAME} & \text{PLOCATION} & \text{HOURS} \\
\hline
R_1 & b_{11} & a_2 & b_{13} & b_{14} & a_5 & b_{16} \\
R_2 & a_1 & b_{22} & a_3 & a_4 & a_5 & a_6
\end{array}$$

(no changes to matrix after applying functional dependencies)

(b)

EMP

SSN	ENAME

PROJECT

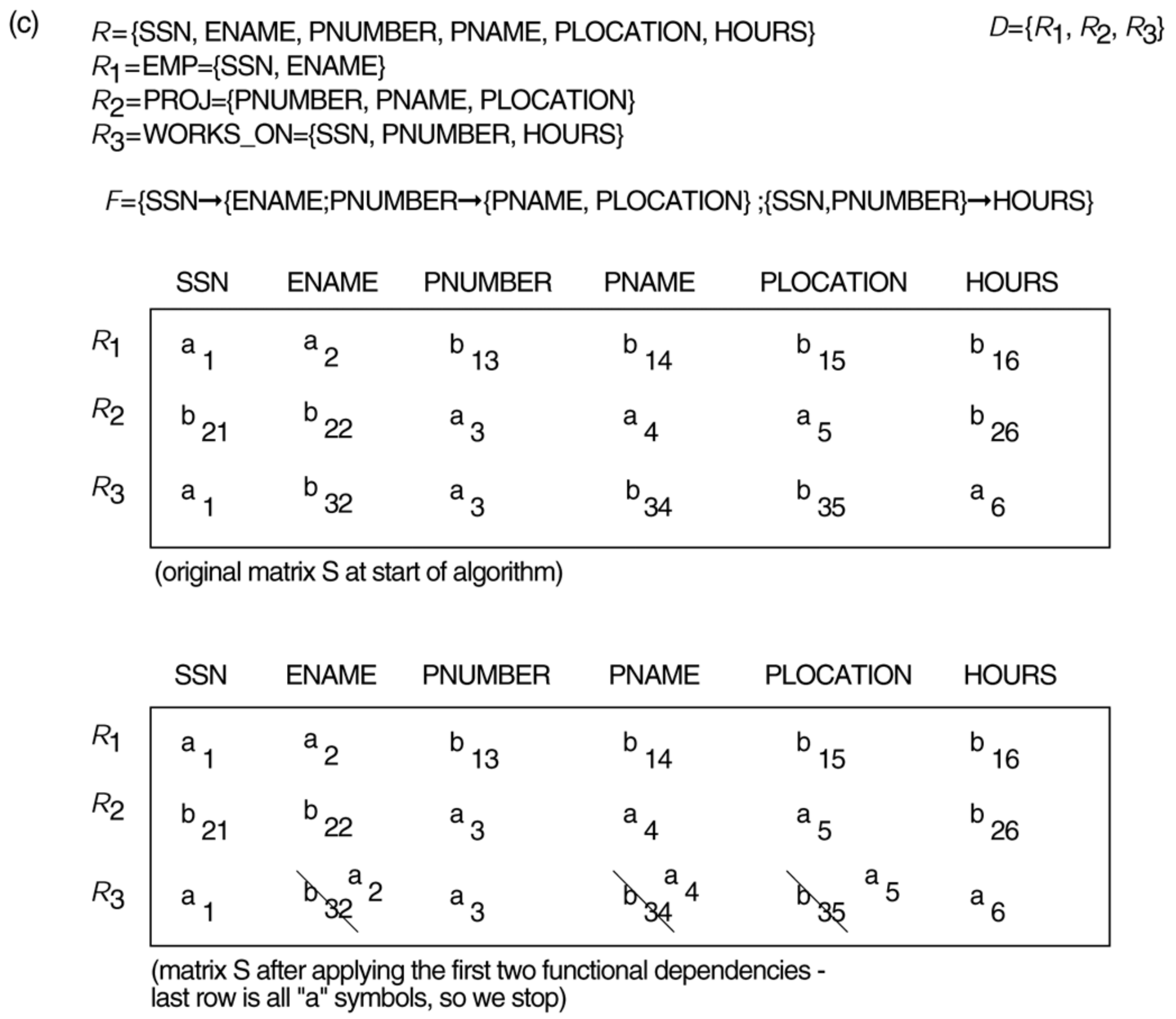
PNUMBER	PNAME	PLOCATION

WORKS_ON

SSN	PNUMBER	HOURS

Properties of Relational Decompositions (8)

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



(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→{ENAME;PNUMBER→{PNAME, PLOCATION} ;{SSN,PNUMBER}→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	$\cancel{b_{32}}$ 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)

Properties of Relational Decompositions (9)

- **Testing Binary Decompositions for Lossless Join Property**
  - **Binary Decomposition:** Decomposition of a relation R into two relations.
  - **PROPERTY LJ1 (lossless 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. $((R1 \cap R2) \rightarrow (R1 - R2))$ is in F^+, or
    - The f.d. $((R1 \cap R2) \rightarrow (R2 - R1))$ is in F^+.

Properties of Relational Decompositions (10)

- **Successive Lossless 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.

2. Algorithms for Relational Database Schema Design (1)

- **Algorithm 11.2: Relational Synthesis into 3NF with Dependency Preservation (Relational Synthesis Algorithm)**
  - **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 10.2);
2. For each left-hand-side X of a functional dependency that appears in G,

   create a relation schema in D with attributes {X ∪ {A1} ∪ {A2} ... ∪ {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. Place any remaining attributes (that have not been placed in any relation) in a single relation schema to ensure the attribute preservation property.
   - **Claim 3: Every relation schema created by Algorithm 11.2 is in 3NF.**

Algorithms for Relational Database Schema Design (2)

- **Algorithm 11.3: 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 ∪ Y);
   };
```

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

Algorithms for Relational Database Schema Design (3)

- **Algorithm 11.4 Relational Synthesis into 3NF with Dependency Preservation and Lossless (Non-Additive) 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 10.2).
2. For each left-hand-side X of a functional dependency that appears in G,

   create a relation schema in D with attributes {X ᴜ {A1} ᴜ {A2} ... ᴜ {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. 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 11.4a to find the key of R)*

Algorithms for Relational Database Schema Design (4)

- **Algorithm 11.4a 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};
   }
```

Algorithms for Relational Database Schema Design (5)

(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 11.2
Issues with NULL-value joins. (a) Some EMPLOYEE tuples have NULL for the join attribute Dnum. (b) Result of applying NATURAL JOIN to the EMPLOYEE and DEPARTMENT relations. (c) Result of applying LEFT OUTER JOIN to EMPLOYEE and DEPARTMENT.

Algorithms for Relational Database Schema Design (5)

(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

Figure 11.2
Issues with NULL-value joins. (a) Some EMPLOYEE tuples have NULL for the join attribute Dnum. (b) Result of applying NATURAL JOIN to the EMPLOYEE and DEPARTMENT relations. (c) Result of applying LEFT OUTER JOIN to EMPLOYEE and DEPARTMENT.

Algorithms for Relational Database Schema Design (6)

Figure 11.3
The dangling tuple problem.
(a) The relation EMPLOYEE_1 (includes all attributes of EMPLOYEE from Figure 11.2(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).

(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

Algorithms for Relational Database Schema Design (6)

Figure 11.3
The dangling tuple problem.
(a) The relation EMPLOYEE_1 (includes all attributes of EMPLOYEE from Figure 11.2(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).

(b) EMPLOYEE_2

<u>Ssn</u>	Dnum
123456789	5
333445555	5
999887777	4
987654321	4
666884444	5
453453453	5
987987987	4
888665555	1
999775555	NULL
888664444	NULL

(c) EMPLOYEE_3

<u>Ssn</u>	Dnum
123456789	5
333445555	5
999887777	4
987654321	4
666884444	5
453453453	5
987987987	4
888665555	1

Algorithms for Relational Database Schema Design (7)

- **Discussion of Normalization Algorithms:**
- Problems:
  - The database designer must first specify *all* the relevant functional dependencies among the database attributes.
  - 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 11.4).

Algorithms for Relational Database Schema Design (8)

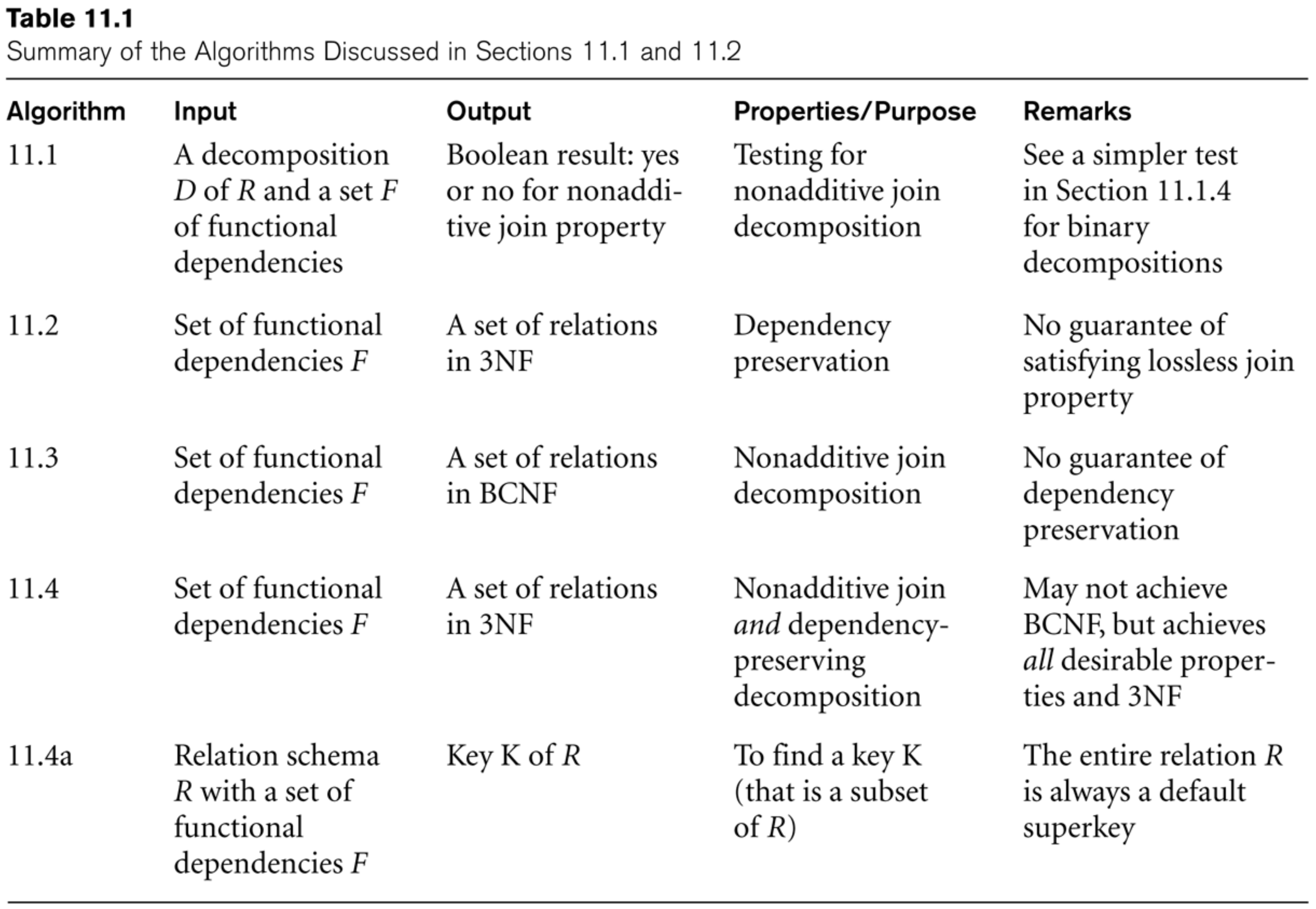


Table 11.1
Summary of the Algorithms Discussed in Sections 11.1 and 11.2

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, but achieves *all* desirable properties and 3NF
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

5. 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

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

- **Note:**
  - The ? (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)

- **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
- **Inclusion dependency inference rules**
  - **IDIR1 (reflexivity)**: $R.X < R.X$.
  - **IDIR2 (attribute correspondence)**: If $R.X < S.Y$
    - where $X = \{A_1, A_2, \dots, A_n\}$ and $Y = \{B_1, B_2, \dots, B_n\}$ and A_i *Corresponds-to* B_i, then $R.A_i < S.B_i$
    - for $1 \leq i \leq n$.
  - **IDIR3 (transitivity)**: If $R.X < S.Y$ and $S.Y < T.Z$, then $R.X < T.Z$.

6. Other Dependencies and Normal Forms (1)

Template Dependencies:

- Template dependencies provide a technique for representing constraints in relations that typically have no easy and formal definitions.
- The idea is to specify a template—or example—that defines each constraint or dependency.
- There are two types of templates:
  - tuple-generating templates
  - constraint-generating templates.
- A template consists of a number of **hypothesis tuples** that are meant to show an example of the tuples that may appear in one or more relations. The other part of the template is the **template conclusion.**

Other Dependencies and Normal Forms (2)

Figure 11.6
Templates for some common type of dependencies.
(a) Template for functional dependency $X \rightarrow Y$.
(b) Template for the multivalued dependency $X \twoheadrightarrow Y$.
(c) Template for the inclusion dependency $R.X < S.Y$.

(a) $R = \{A, B, C, D\}$

$X = \{A, B\}$
$Y = \{C, D\}$

Hypothesis

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2

Conclusion $c_1 = c_2$ and $d_1 = d_2$

(b) $R = \{A, B, C, D\}$

$X = \{A, B\}$
$Y = \{C\}$

Hypothesis

A	B	C	D
a_1	b_1	c_1	d_1
a_1	b_1	c_2	d_2

Conclusion

A	B	C	D
a_1	b_1	c_2	d_1
a_1	b_1	c_1	d_2

(c) $R = \{A, B, C, D\}$ $S = \{E, F, G\}$

$X = \{C, D\}$
$Y = \{E, F\}$

Hypothesis

A	B	C	D
a_1	b_1	c_1	d_1

Conclusion

E	F	G
c_1	d_1	g

Other Dependencies and Normal Forms (3)

Figure 11.7
Templates for the constraint that an employee's salary must be less than the supervisor's salary.

EMPLOYEE = {Name, Ssn, . . . , Salary, Supervisor_ssn}

	a	b	c	d
Hypothesis	e	d	f	g
Conclusion			$c < f$	

Other Dependencies and Normal Forms (4)

Domain-Key Normal Form (DKNF):

- **Definition:**
  - A relation schema is said to be in **DKNF** if all constraints and dependencies that should hold on the valid relation states can be enforced simply by enforcing the domain constraints and key constraints on the relation.
- The **idea** is to specify (theoretically, at least) the "*ultimate normal form*" that takes into account all possible types of dependencies and constraints. .
- For a relation in DKNF, it becomes very straightforward to enforce all database constraints by simply checking that each attribute value in a tuple is of the appropriate domain and that every key constraint is enforced.
- The practical utility of DKNF is limited

Questions

1. Explain the informal design guidelines for the database design.
2. Which normal form is based on full functional dependency? Explain the normal form which is based on this.
3. What is transitive dependency? Explain 3NF with example.
4. What is multivalued dependency? Explain 4NF with example.
5. Write an algorithm to find the minimal cover.