FACULTY OF COMPUTER SCIENCE AND ENGINEERING

FUNCTIONAL DEPENDENCIES AND NORMALIZATION

DATABASES - lectures

Outline

- Informal Design Guidelines for Relational Databases
- Functional Dependencies (FDs)
- Normal Forms (NFs) for Relational Databases
  - 1NF
  - 2NF
  - 3NF and BCNF
  - 4NF
  - 5NF

Informal Design Guidelines for Relational Databases

- 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
- Informal criteria for “good” base relations:
  - Making sure that the semantics of the attributes is clear in the schema
  - Reducing the redundant information in tuples
  - Reducing the NULL values in tuples
  - Disallowing the possibility of generating spurious tuples

Semantics of the Relation Attributes

- **GUIDELINE 1** : Informally, each tuple in a relation should represent one entity or relationship instance.
  - 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.*

Example

EMPLOYEE F.K.

Ename	<u>Ssn</u>	Bdate	Address	Dnumber

P.K.

DEPARTMENT F.K.

Dname	<u>Dnumber</u>	Dmgr_ssn

P.K.

DEPT_LOCATIONS
F.K.

<u>Dnumber</u>	<u>Dlocation</u>

P.K.

PROJECT F.K.

Pname	<u>Pnumber</u>	Plocation	Dnum

P.K.

WORKS_ON
F.K. F.K.

<u>Ssn</u>	<u>Pnumber</u>	Hours

P.K.

Figure 10.1
A simplified COMPANY relational database schema.

Figure 10.2
Example database state for the relational database schema of Figure 10.1.

EMPLOYEE

Ename	<u>Ssn</u>	Bdate	Address	Dnumber
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	291Berry, 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

DEPARTMENT

Dname	<u>Dnumber</u>	Dmgr_ssn
Research	5	333445555
Administration	4	987654321
Headquarters	1	888665555

DEPT_LOCATIONS

<u>Dnumber</u>	<u>Dlocation</u>
1	Houston
4	Stafford
5	Bellaire
5	Sugarland
5	Houston

WORKS_ON

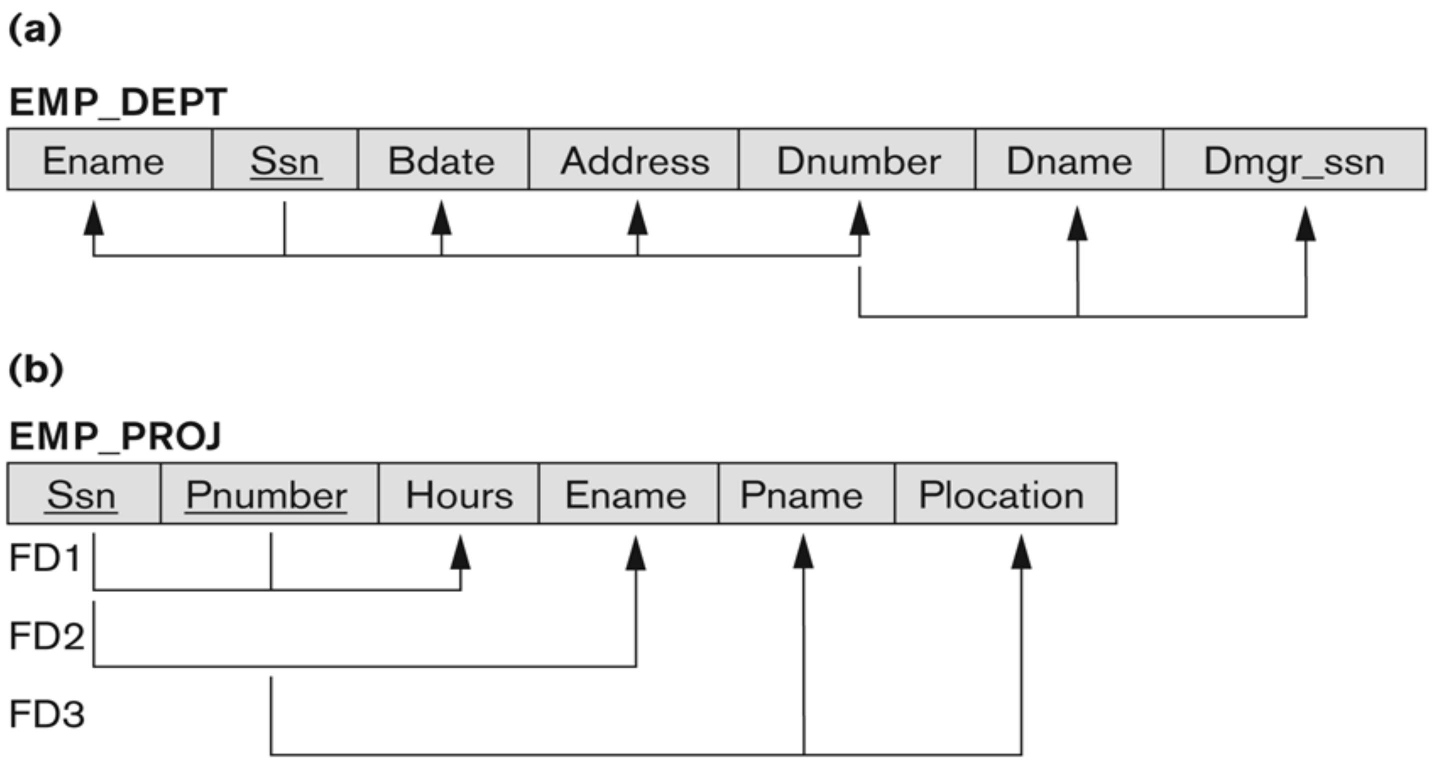
<u>Ssn</u>	<u>Pnumber</u>	Hours
123456789	1	32.5
123456789	2	7.5
666884444	3	40.0
453453453	1	20.0
453453453	2	20.0
333445555	2	10.0
333445555	3	10.0
333445555	10	10.0
333445555	20	10.0
999887777	30	30.0
999887777	10	10.0
987987987	10	35.0
987987987	30	5.0
987654321	30	20.0
987654321	20	15.0
888665555	20	Null

PROJECT

Pname	<u>Pnumber</u>	Plocation	Dnum
ProductX	1	Bellaire	5
ProductY	2	Sugarland	5
ProductZ	3	Houston	5
Computerization	10	Stafford	4
Reorganization	20	Houston	1
Newbenefits	30	Stafford	4

A simplified COMPANY relational database schema

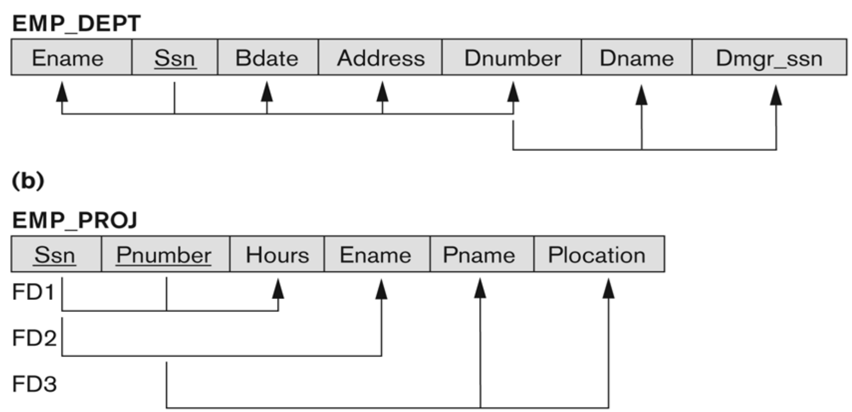
Easy to Explain Its Meaning



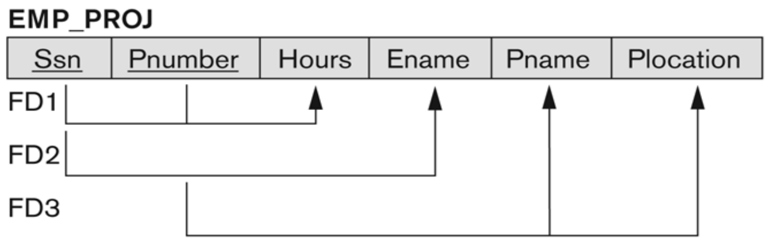
Redundant Information in Tuples

- Information is stored redundantly
  - Wastes storage
  - Causes problems with update anomalies
    - Insertion anomalies
    - Deletion anomalies
    - Modification anomalies

Example of redundant information in tuples



Example – modification anomalies



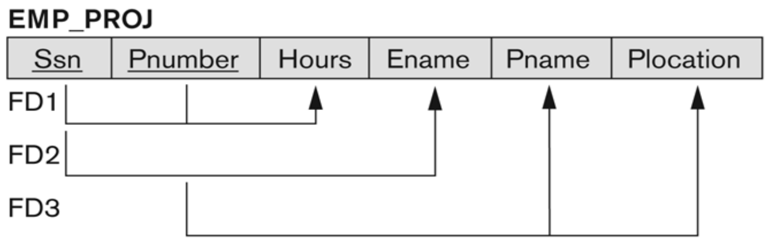
- Changing the name of project number P1 from “Billing” to “Customer-Accounting” may cause this update to be made for all 100 employees working on project P1.

Redundancy (Ename) Redundancy (Pname, Plocation)

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

Example – insertion anomalies



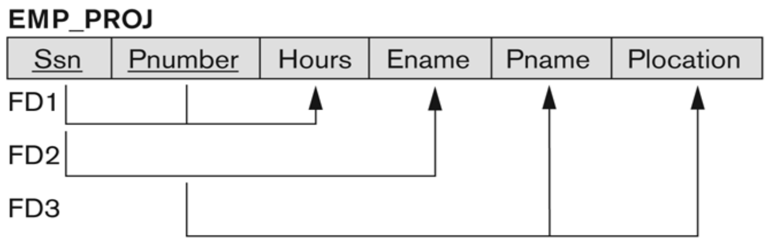
- Cannot insert a project unless an employee is assigned to it.
- Cannot insert an employee unless he/she is assigned to a project.

Redundancy (Ename) — Redundancy (Pname, Plocation)

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

Example – deletion anomalies



- 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.

Redundancy Redundancy

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

Redundant Information in Tuples

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

Null Values in Tuples

- **<u>GUIDELINE 3</u>** :
  - 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
- **<u>GUIDELINE 4</u>** :
  - 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

- 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.
  - *Design relation schemas so that they can be* ***joined with equality conditions on attributes that are appropriately related*** *(primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated.*

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.
  - *Design relation schemas so that they can be* ***joined with equality conditions on attributes that are appropriately related*** *(primary key, foreign key) pairs in a way that guarantees that no spurious tuples are generated*
  - ***Avoid relations that contain matching attributes that are not (foreign key, primary key) combinations*** *because joining on such attributes may produce spurious tuples*

Example

EMP_LOCS

Ename	Plocation

P.K.

EMP_PROJ1

Ssn	Pnumber	Hours	Pname	Plocation

P.K.

EMP_LOCS * EMP_PROJ1=?

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

Example

	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.

*
*
*

Functional Dependencies

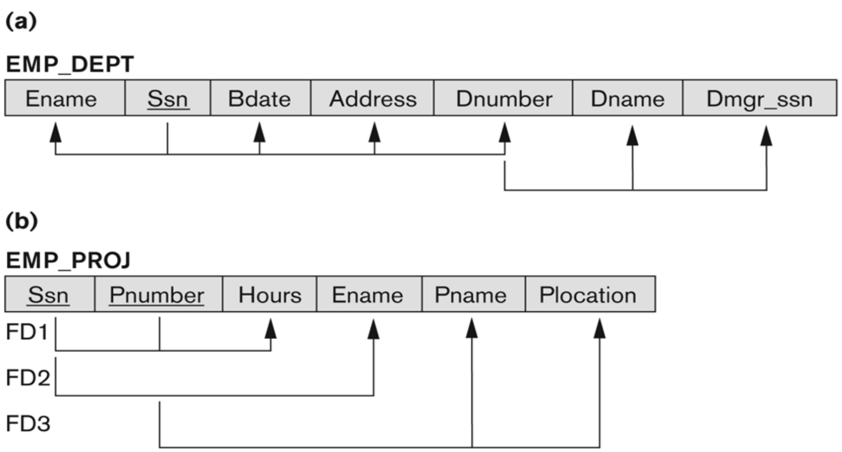
- Functional Dependencies (FDs)
  - Are used to specify *formal measures* of the "goodness" of relational designs
  - And keys are used to define **normal forms** for relations
  - Are **constraints** that are derived from the *meaning* and *interrelationships* of the data attributes

- FDs are derived from the real-world constraints on the attributes

Functional Dependencies

- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y
  - Denoted by X → Y
- X → Y holds iff 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)

Examples of FD constraints



- Social security number determines employee name

  $SSN \rightarrow ENAME$

- Project number determines project name and location

  $PNUMBER \rightarrow \{PNAME, PLOCATION\}$

- Employee ssn and project number determines the hours per week that the employee works on the project

  $\{SSN, PNUMBER\} \rightarrow HOURS$

Example

- A functional dependency is a property of the **semantics or meaning of the attributes for all relation states.**

- The database designers will use their understanding of the semantics of the attributes of R—that is, how they relate to one another—to specify the functional dependencies that should hold on **all** relation states (extensions) r of R

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

Figure 10.7
A relation state of TEACH with a *possible* functional dependency TEXT → COURSE. However, TEACHER → COURSE is ruled out.

Example

- A functional dependency is a property of the **semantics or meaning of the attributes for all relation states.**

- The database designers will use their understanding of the semantics of the attributes of R—that is, how they relate to one another—to specify the functional dependencies that should hold on **all** relation states (extensions) r of R

TEACH

Teacher	Course	Text
Smith	Data Structures	Bartram
Smith	Data Management	Martin
Hall	Compilers	Hoffman
Brown	Data Structures	Horowitz

Figure 10.7
A relation state of TEACH with a *possible* functional dependency TEXT → COURSE. However, TEACHER → COURSE is ruled out.

Inference Rules for FDs

- 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
  - All dependencies inferred using these rules hold in all relation states
  - All possible dependencies can be inferred using these rules repeatedly

Inference Rules for FDs

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

a set of attributes always determines itself or any of its subsets

- 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
  - All dependencies inferred using these rules hold in all relation states
  - All possible dependencies can be inferred using these rules repeatedly

Inference Rules for FDs

- 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

adding the same set of attributes to both the left- and right-hand sides of a dependency results in another valid dependency

- IR1, IR2, IR3 form a **sound** and **complete** set of inference rules
  - All dependencies inferred using these rules hold in all relation states
  - All possible dependencies can be inferred using these rules repeatedly

Inference Rules for FDs

- 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)

Inference Rules for FDs

- 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

we can remove attributes from
the right-hand side of a dependency

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

Inference Rules for FDs

- 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

we can combine a set of dependencies
Into a single FD

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

Inference Rules for FDs

- 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

We can replace a set of attributes *Y* on the left hand side of a dependency with another set *X* that functionally determines *Y*

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

Inference Rules for FDs

- 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

    Cautionary note:
    XY → A does *not* necessarily imply either *X → A* or *Y → A*.

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

Inference Rules for FDs

- **Closure** of a set F of FDs is the set F^+ of all FDs that can be inferred from F, including F
- **Closure** of a set of attributes X **with respect to** F is the set X^+ of all attributes that are functionally determined by X
- X^+ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F

Equivalence of Sets of FDs

- Two sets of FDs F and G are **equivalent** if:
  - Every FD in F can be inferred from G, and
  - Every FD in G can be inferred from F
  - Hence, F and G are equivalent if $F^+ = G^+$
- Definition (**Covers**):
  - F **covers** G if every FD in G can be inferred from F
    - (i.e., if G^+ *subset-of* F^+)
- F and G are equivalent if F covers G and G covers F
- There is an algorithm for checking equivalence of sets of FDs

Minimal Sets of FDs

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

- 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.*

Minimal Sets of FDs

- A set of FDs is **minimal** if it satisfies the following conditions:
  1. Every dependency in F has a single attribute for its right-hand side.
  2. We cannot remove any dependency from F and have a set of dependencies that is equivalent to F.
  3. Ensures that there are no redundancies in the dependencies either by having redundant attributes on the left-hand side of a dependency

- 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.*

Minimal Sets of FDs

- A set of FDs is **minimal** if it satisfies the following conditions:
  1. Every dependency in F has a single attribute for its right-hand side.
  2. Ensures that there are no redundancies in the dependencies either by having a dependency that can be inferred from the remaining FDs in *F*
  3. We cannot replace any dependency X → A in F with a dependency Y → A, where Y proper-subset-of X (Y subset-of X) and still have a set of dependencies that is equivalent to F.
- 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 for Finding a Minimal Cover *F* for a Set of Functional Dependencies *E*

1. Set F := E.

2. Replace each functional dependency $X \rightarrow \{A_1, A_2, \ldots, A_n\}$ in F by the *n* functional dependencies $X \rightarrow A_1$, $X \rightarrow A_2$, ... , $X \rightarrow A_n$.

3. For each functional dependency $X \rightarrow A$ in F
   For each attribute *B* that is an element of X
   if $\{ \{F - \{X \rightarrow A\} \} \cup \{ (X - \{B\}) \rightarrow A\} \}$ is equivalent to F
   then replace $X \rightarrow A$ with $(X - \{B\}) \rightarrow A$ in F.

4. For each remaining functional dependency $X \rightarrow A$ in F
   if $\{ F - \{X \rightarrow A\} \}$ is equivalent to F
   then remove $X \rightarrow A$ from F.

Example

Let the given set of FDs be E : {B → A, D → A, AB → D}.We have to find the minimum cover of E.

All above dependencies are in canonical form; so we have completed step 1 of the 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. 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.

Hence the minimum cover of E is **{B → D, D → A}**.

Normalization of Relations

- **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
- **Denormalization**:
  - The process of storing the join of higher normal form relations as a base relation—which is in a lower normal form

Normalization of Relations

- 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

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)

Definitions of Keys and Attributes Participating in Keys

- A **superkey** of a relation schema R = {A1, A2, ..., An} is a set of attributes S *subset-of* R with the property that no two tuples t1 and t2 in any legal relation state r of R will have t1[S] = t2[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

- 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 1NF

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

- Considered to be part of the definition of relation

Example

Normalization into 1NF

(a)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocations

(b)

DEPARTMENT

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

(c)

DEPARTMENT

Dname	Dnumber	Dmgr_ssn	Dlocation
Research	5	333445555	Bellaire
Research	5	333445555	Sugarland
Research	5	333445555	Houston
Administration	4	987654321	Stafford
Headquarters	1	888665555	Houston

Figure 10.8
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Example state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

Can you think of any other solutions to normalize DEPARTMENT?

Example

Normalizing nested relations into 1 NF

(a)

EMP_PROJ

Ssn	Ename	Projs	
		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, AliciaJ.	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 10.9
Normalizing nested relations into 1NF. (a) Schema of the EMP_PROJ relation with a *nested relation* attribute PROJS. (b) Example 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.

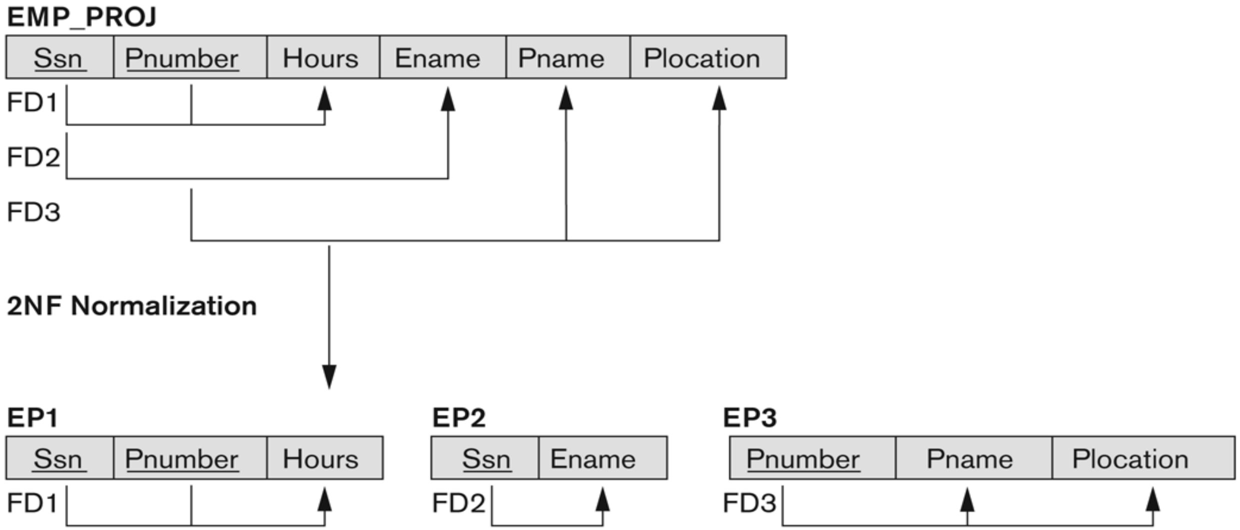
Second Normal Form 2NF

- Uses the concepts of **FDs, primary key**
- Definitions
  - **Prime attribute:** An attribute that is member of the primary key K
  - **Full functional dependency:** a FD $Y \rightarrow Z$ where removal of any attribute from Y means the FD does not hold any more
- 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

Example

Normalizing into 2NF

- {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



Third Normal Form 3NF

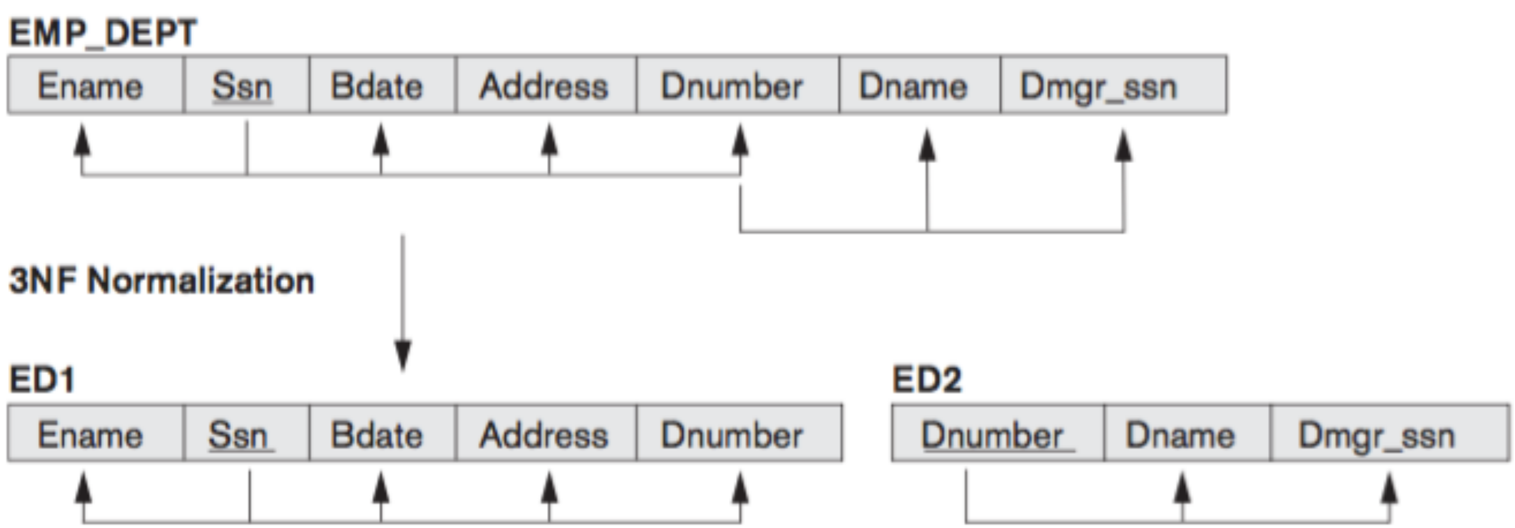
- Definition:
  - **Transitive functional dependency:** a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$
- A relation schema R is in **third normal form (3NF)** if it is in 2NF *and* no non-prime attribute A in R is *transitively dependent on the primary key*
- NOTE:
  - In $X \rightarrow Y$ and $Y \rightarrow Z$, with X as the primary key, we consider this a problem only if Y is not 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.

Third Normal Form 3NF

- Definition:
  - **Transitive functional dependency:** a FD X → Y that can be derived from two FDs X → Z and Z → Y
- *There exists a set of attributes Z in R that is neither a candidate key nor a subset of any key of R*
- A relation schema R is in **third normal form (3NF)** if it is in 2NF *and* no non-prime attribute A in R is transitively dependent on the primary key

Third Normal Form 3NF

- 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



Normal Forms Defined Informally

- 1st normal form
  - All attributes depend on **the key**
- 2nd normal form
  - All attributes depend on **the whole key**
- 3rd normal form
  - All attributes depend on **nothing but the key**

SUMMARY OF NORMAL FORMS based on Primary Keys

Normal Form	Test	Remedy (Normalization)
First (1NF)	Relation should have no multivalued attributes or nested relations.	Form new relations for each multivalued attribute or nested relation.
Second (2NF)	For relations where primary key contains multiple attributes, no nonkey attribute should be functionally dependent on a part of the primary key.	Decompose and set up a new relation for each partial key with its dependent attribute(s). Make sure to keep a relation with the original primary key and any attributes that are fully functionally dependent on it.
Third (3NF)	Relation should not have a nonkey attribute functionally determined by another nonkey attribute (or by a set of nonkey attributes). That is, there should be no transitive dependency of a nonkey attribute on the primary key.	Decompose and set up a relation that includes the nonkey attribute(s) that functionally determine(s) other nonkey attribute(s).

General Normal Form Definitions (For Multiple Keys)

- 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

- 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

if every nonprime attribute of *R* meets both of the following conditions:

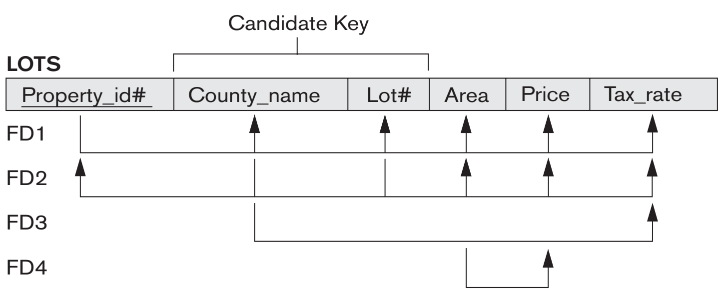
It is fully functionally dependent ON EVERY KEY of *R*.

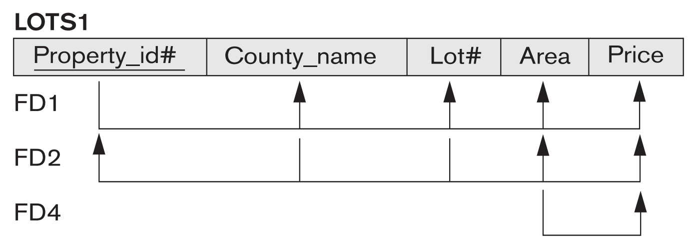
It is nontransitively dependent ON EVERY KEY of *R*.

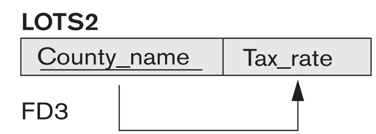
- NOTE: Boyce-Codd normal form disallows condition (b) above

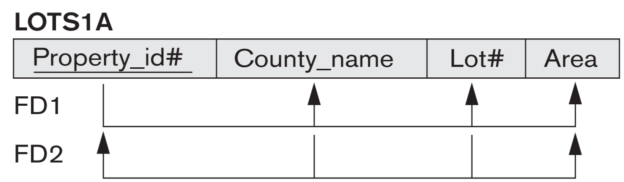
Example

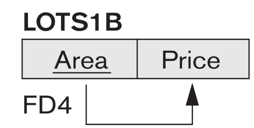
Successive Normalization of LOTS into 2NF and 3NF

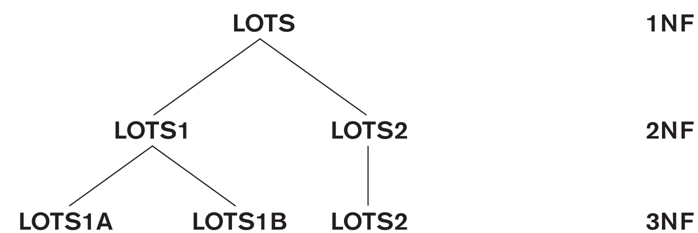






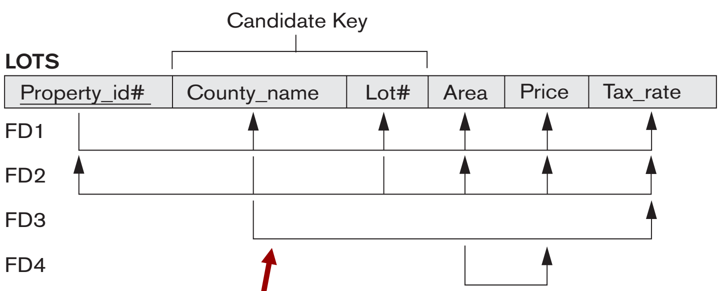




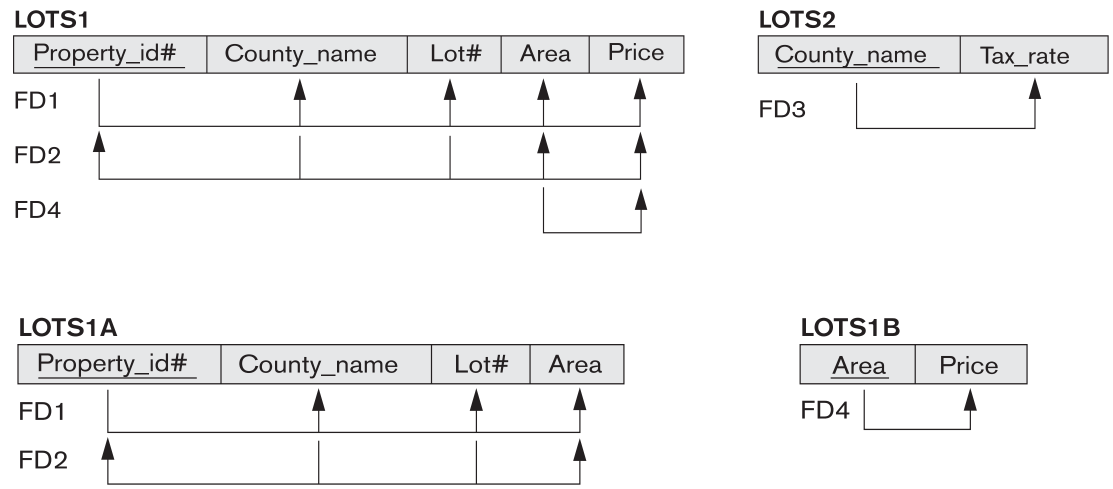


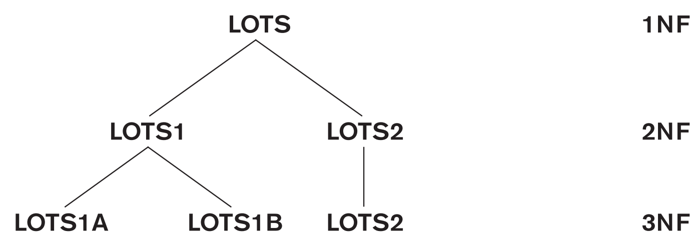
Example

Successive Normalization of LOTS into 2NF and 3NF



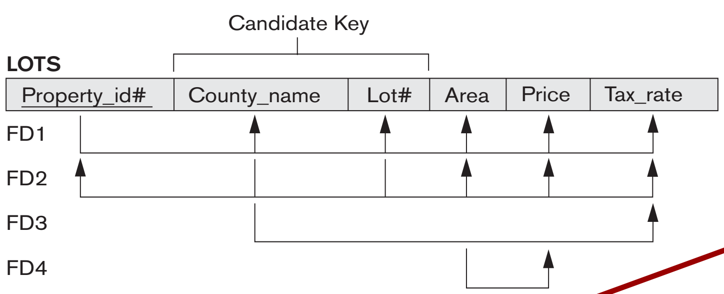
violates the general definition of 2NF because Tax_rate is partially dependent on the candidate key {County_name, Lot#}

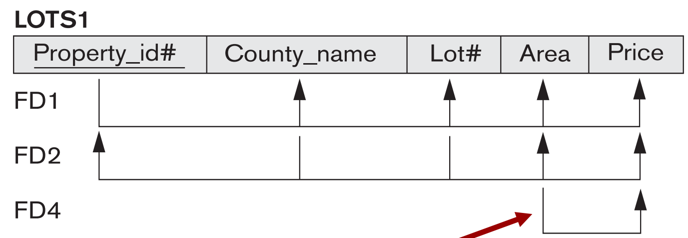


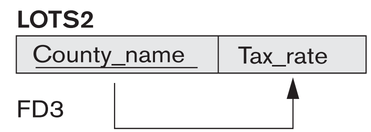


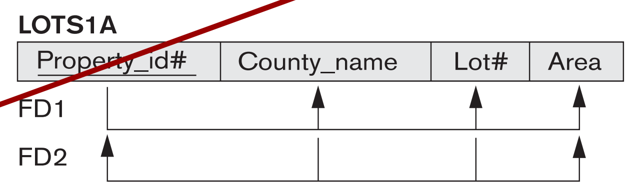
Example

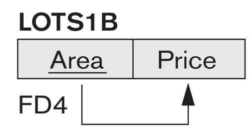
Successive Normalization of LOTS into 2NF and 3NF



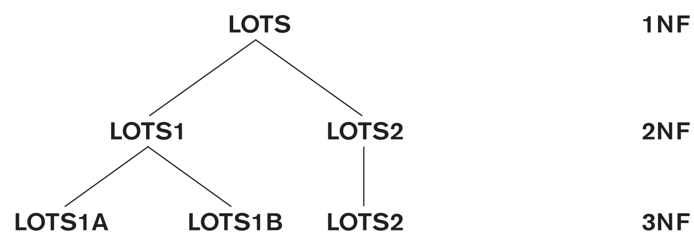






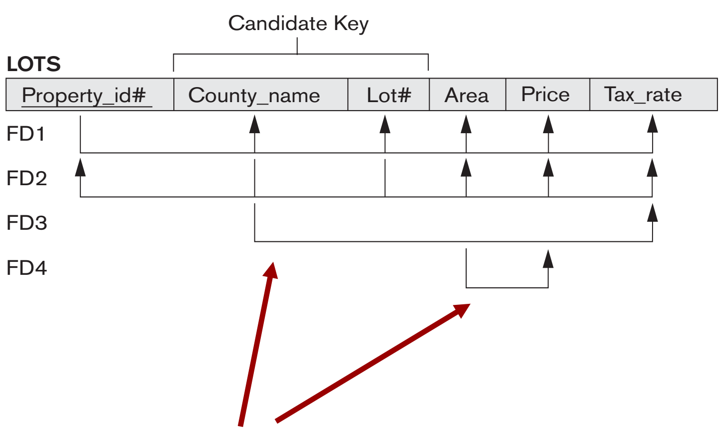


FD4 in LOTS1 violates 3NF because Area is not a superkey and Price is not a prime attribute in LOTS1.

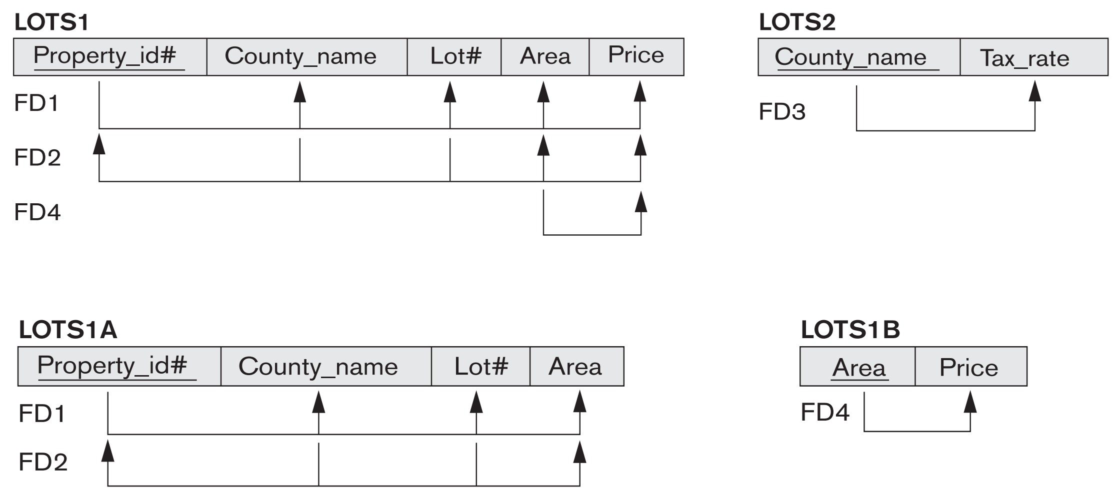


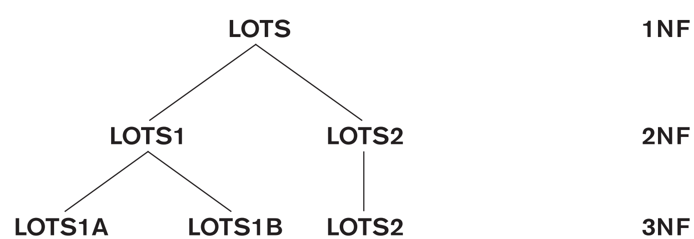
Example

Successive Normalization of LOTS into 2NF and 3NF



The transitive and partial dependencies that violate 3NF can be removed *in any order*





BCNF (Boyce-Codd Normal Form)

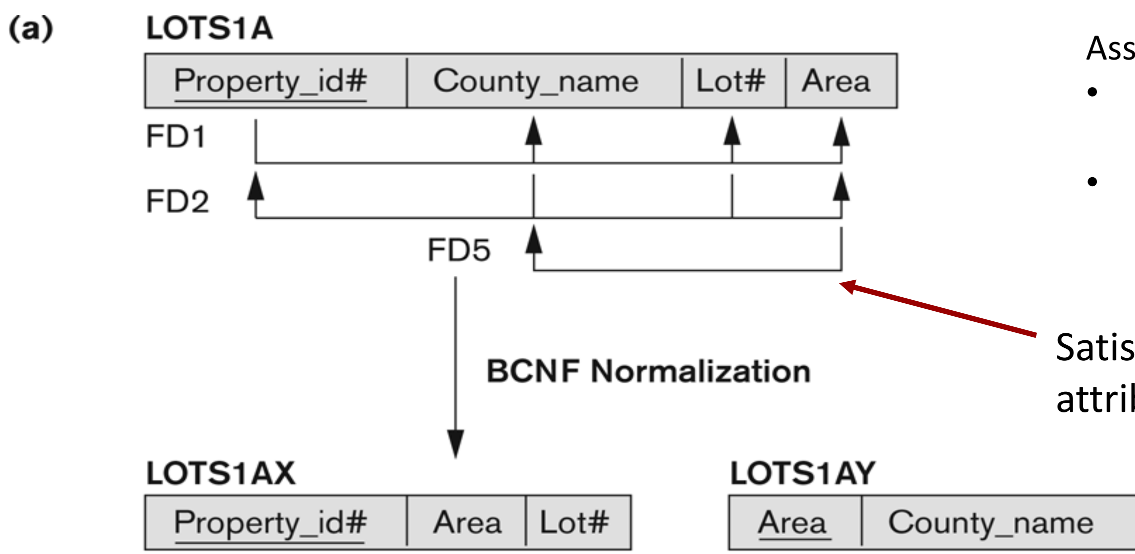
- A relation schema R is in **Boyce-Codd Normal Form (BCNF)** if whenever an **FD X → A** holds in R, then **X is a superkey** of R

- 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)

Example



Assume

- **DeKalb County** are **only** **0.5, 0.6, 0.7, 0.8, 0.9, and 1.0** acres
- **Fulton County** are restricted to **1.1, 1.2, ..., 1.9, and 2.0 acres**

Satisfies 3NF since County_name is prime attribute

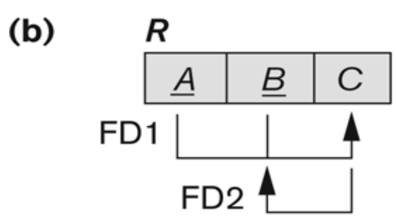
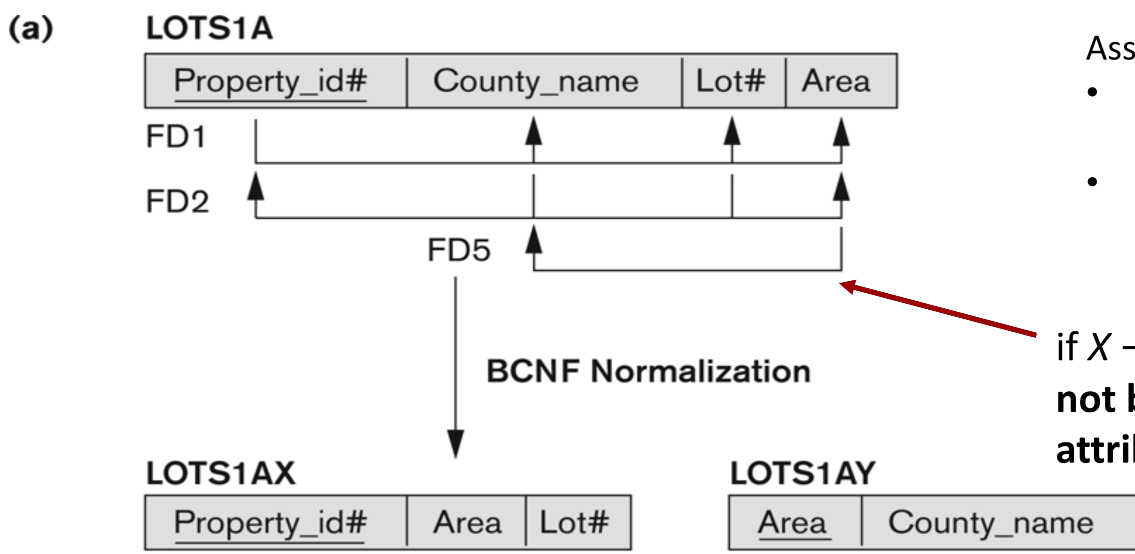


Figure 10.12
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.

Example



Assume

- **DeKalb County** are **only** **0.5, 0.6, 0.7, 0.8, 0.9, and 1.0** acres
- **Fulton County** are restricted to **1.1, 1.2, ..., 1.9, and 2.0 acres**

if $X \rightarrow A$ holds in a relation schema R with ***X* not being a superkey** *and* ***A* being a prime attribute**, *R will* be in 3NF but not in BCNF

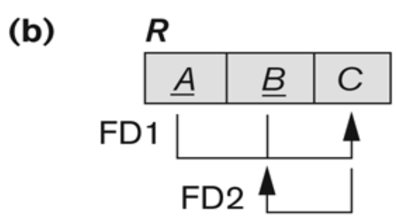


Figure 10.12
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.

Example

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

FD2

TEACH

Student	Course	Instructor
Narayan	Database	Mark
Smith	Database	Navathe
Smith	Operating Systems	Ammar
Smith	Theory	Schulman
Wallace	Database	Mark
Wallace	Operating Systems	Ahamad
Wong	Database	Omiecinski
Zelaya	Database	Navathe
Narayan	Operating Systems	Ammar

FD1

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

Achieving the BCNF by Decomposition

- Three possible decompositions for relation TEACH
  - {<u>student, instructor</u>} and {<u>student, course</u>}
  - {course, <u>instructor</u> } and {<u>course, student</u>}
  - {<u>instructor</u>, course } and {<u>instructor, student</u>}
- 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).

Multivalued Dependencies

Definition:

X multidetermines *Y*

- A **multivalued dependency (MVD)** $X \twoheadrightarrow Y$ specified on relation schema R, where X and Y are both subsets of R, specifies the following constraint on any relation state r of R: If two tuples t1 and t2 exist in r such that t1[X] = t2[X], then two tuples t3 and t4 should also exist in r with the following properties, where we use Z to denote (R – (X U Y)):
  - t3[X] = t4[X] = t1[X] = t2[X].
  - t3[Y] = t1[Y] and t4[Y] = t2[Y].
  - t3[Z] = t2[Z] and t4[Z] = t1[Z].
- An MVD $X \twoheadrightarrow Y$ in R is called **a trivial MVD** if (a) Y is a subset of X, or (b) X U Y = R.

Multivalued Dependencies

Ename ↠ Pname

Ename ↠ Dname

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

Multivalued Dependencies

- **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**): $\{X \twoheadrightarrow Y, Y \twoheadrightarrow Z\} \models X \twoheadrightarrow (Z - Y)$.
  - 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 $X \rightarrow Z$.

Armstrong's rules

MVD-related

Relate FD and MVD

Fourth Normal Form 4NF

Definition:

- A relation schema *R* is in **4NF** with respect to a set of dependencies *F* (that includes functional dependencies and multivalued dependencies) if, for every *nontrivial* multivalued dependency $X \twoheadrightarrow Y$ in F^+, *X* is a superkey for R.

- Reminder: 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 the **closure** of *F*.

Example

Figure 16.4
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.

Ename $\twoheadrightarrow$ Pname

Ename $\twoheadrightarrow$ Dname

(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

Lossless (Non-additive) Join Decomposition into 4NF Relations

- Whenever we decompose a relation schema *R* into $R_1 = (X \cup Y)$ and $R_2 = (R - Y)$ based on an MVD $X \twoheadrightarrow Y$ that holds in R, the decomposition has the nonadditive join property

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

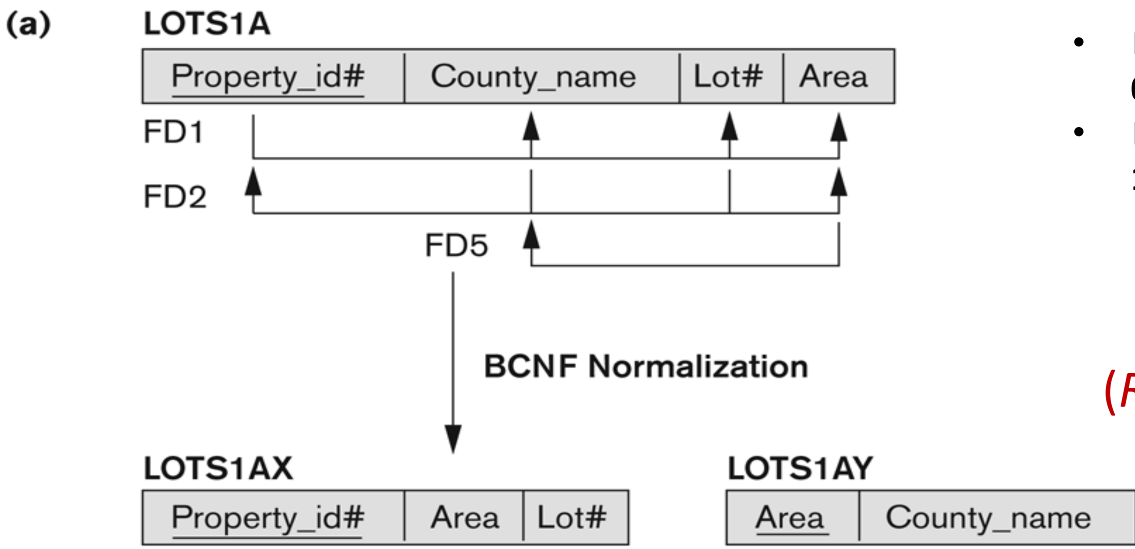
Lossless (Non-additive) Join Decomposition into 4NF Relations

- Whenever we decompose a relation schema *R* into $R_1 = (X \cup Y)$ and $R_2 = (R - Y)$ based on an MVD $X \twoheadrightarrow Y$ that holds in R, the decomposition has the nonadditive join property

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

Decomposed TECH relation into {instructor, course } and {instructor, student}

Example



- **DeKalb County** are **only** **0.5, 0.6, 0.7, 0.8, 0.9, and 1.0** acres
- **Fulton County** are restricted to **1.1, 1.2, ..., 1.9, and 2.0 acres**

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

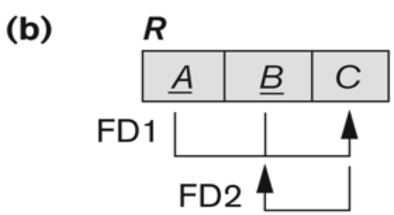


Figure 10.12
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF.

Multivalued Dependencies and 4NF

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

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

$$*(\pi_{R1}(r), \pi_{R2}(r)\ldots, \pi_{Rn}(r)) = r$$

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.

Fifth Normal Form 5NF

Definition:

- 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*.

Example

(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.

No nonadditive join decomposition of *R* into *two* relation schemas, but there may be a nonadditive join decomposition into *more than two* relation schemas

(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

Example

(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

Multiway decomposition into 5NF

(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

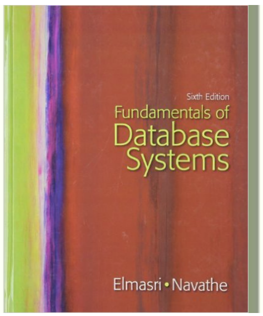
Short summary

- Informal Design Guidelines for Relational Databases
- Functional Dependencies (FDs)
- Normal Forms Based on Primary Keys
- General Normal Form Definitions (For Multiple Keys)
- BCNF (Boyce-Codd Normal Form)
- Multivalued Dependencies
- 4NF, 5NF

Possible exam questions

- Identify FDs by applying the inference rules
- Find the minimal cover for a given set of FDs
- For a given table, keys, and set of FDs check if the relation schema is in a required normal form (with explanation)
- Normalize a given relation schema/table

Bibliography



- **Chapter 15**
- **Chapter 16**

- **Chapter 3**