Database Management Systems

Informal Design Guidelines for Schemas

Prof. Ruby Dinakar & Prof. Nivedita

Department of Computer Science and Engineering

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the below resource and persons:
- Author slides from “Fundamentals of Database Systems”, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

Database Management Systems

Chapter 14: Outline

14.1 Informal Design Guidelines for Relational Databases

14.2 Functional Dependencies (FDs)

14.3 Normal Forms Based on Primary Keys

14.4 General Normal Form Definitions (For Multiple Keys)

14. 5 BCNF (Boyce-Codd Normal Form)

14.6 Multivalued Dependency and Fourth Normal Form

14.7 Join Dependencies and Fifth Normal Form

Database Management Systems

Chapter 15: Outline

15.1 Further topics in Functional Dependencies: Inference Rules, Equivalence and Minimal Cover

15.2 Properties of Relational Decomposition

15.3 Algorithms for Relational Database Schema Design

15.4 About Nulls, Dangling Tuples and Alternative Relational Designs

Database Management Systems

Design Theory

- A relation schema is formed by grouping of attributes.
- A good database design can be achieved by deciding which attributes should be grouped in to a relation to provide better performance.
- There are two levels at which the goodness of relation schemas can be discussed”
  - The logical (conceptual) level
  - The implementation (base relation-storage) level
- Bad design leads to the following problem

  Redundancy

  Inability to represent certain information

  Waste of storage
- Redundancy is the root cause of several problems:

  Redundant storage, insert/delete/update anomalies

  Integrity constraints, in particular functional dependencies, can be used to identify schemas with such problems and to suggest schema refinements.

Database Management Systems

Design Theory contd..

- Decomposition of a relation schema can eliminate redundancy but it also causes problems.
- Problems related to decomposition:
    - Lossless vs. lossy decomposition
    - Dependency preserving vs non dependency preserving decomposition
- Schema refinement aims at addressing these problems by proposing several 'Normal forms'.
- The Schema Refinement divides larger tables to smaller tables and links them using relationships.
- Normalization is a "process of organizing the data in database to avoid data redundancy, insertion anomaly, update anomaly & deletion anomaly".

Informal Design Guidelines for Relational Databases (cont.)

Informal guidelines that may be used as measures to determine the quality of relation schema design:

- 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 Relational Attributes must be clear

- 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.*

- It is assumed that attributes belonging to one relation have certain real-world meaning and a proper interpretation associated with them. The semantics of a relation refers to its meaning resulting from the interpretation of attribute values in a tuple

Database Management Systems

A simplified COMPANY relational database schema – good design

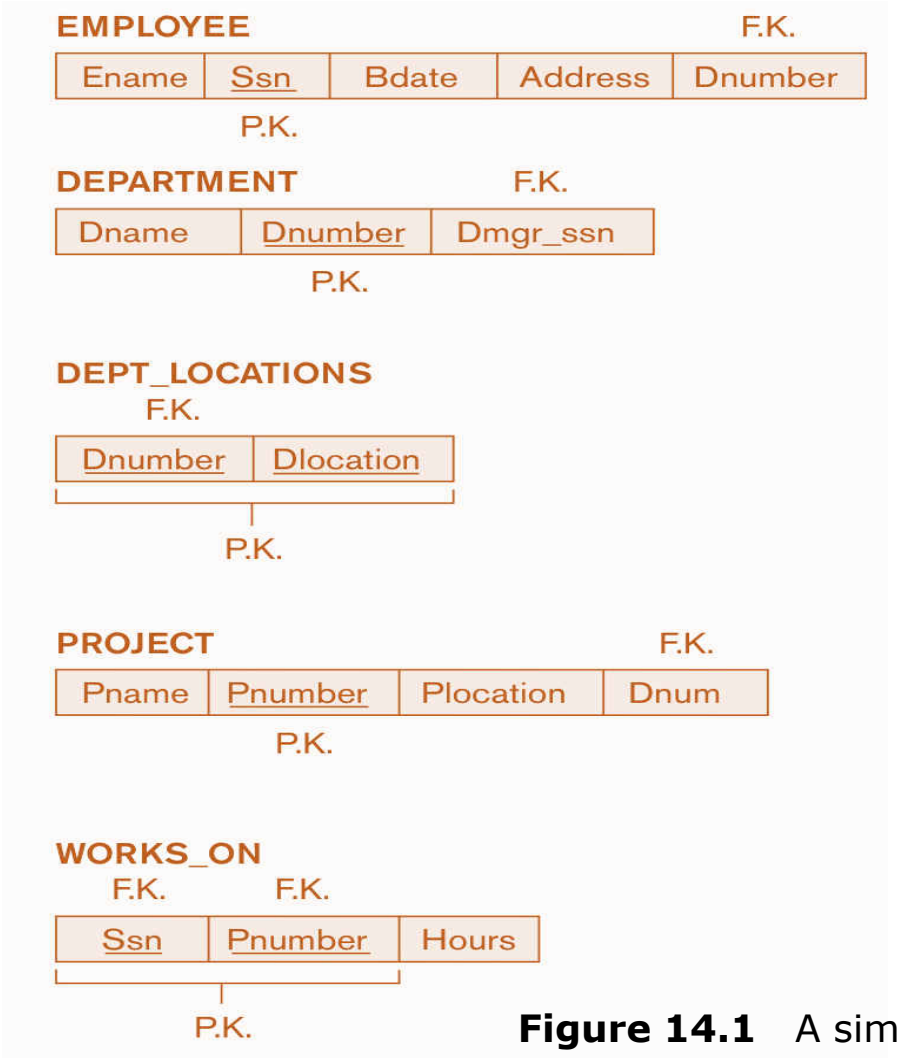


Figure 14.1 A simplified COMPANY relational database schema.

Semantics of the Relational Attributes must be clear

Examples of Violating Guideline 1

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	Dname	Dmgr_ssn

Mixes attributes of employees and departments

EMP_PROJ

Ssn	Pnumber	Hours	Ename	Pname	Plocation

Mixes attributes of employees and projects and the WORKS_ON relationship

EXAMPLE OF AN UPDATE ANOMALY

- Consider the relation:
  - EMP_PROJ(Emp#, Proj#, 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 INSERT ANOMALY

- Consider the relation:
  - EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)
- Insert Anomaly:
  - Cannot insert a project unless an employee is assigned to it.
- Conversely
  - Cannot insert an employee unless an he/she is assigned to a project.

EXAMPLE OF A DELETE ANOMALY

- Consider the relation:
  - EMP_PROJ(Emp#, Proj#, Ename, Pname, No_hours)
- 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

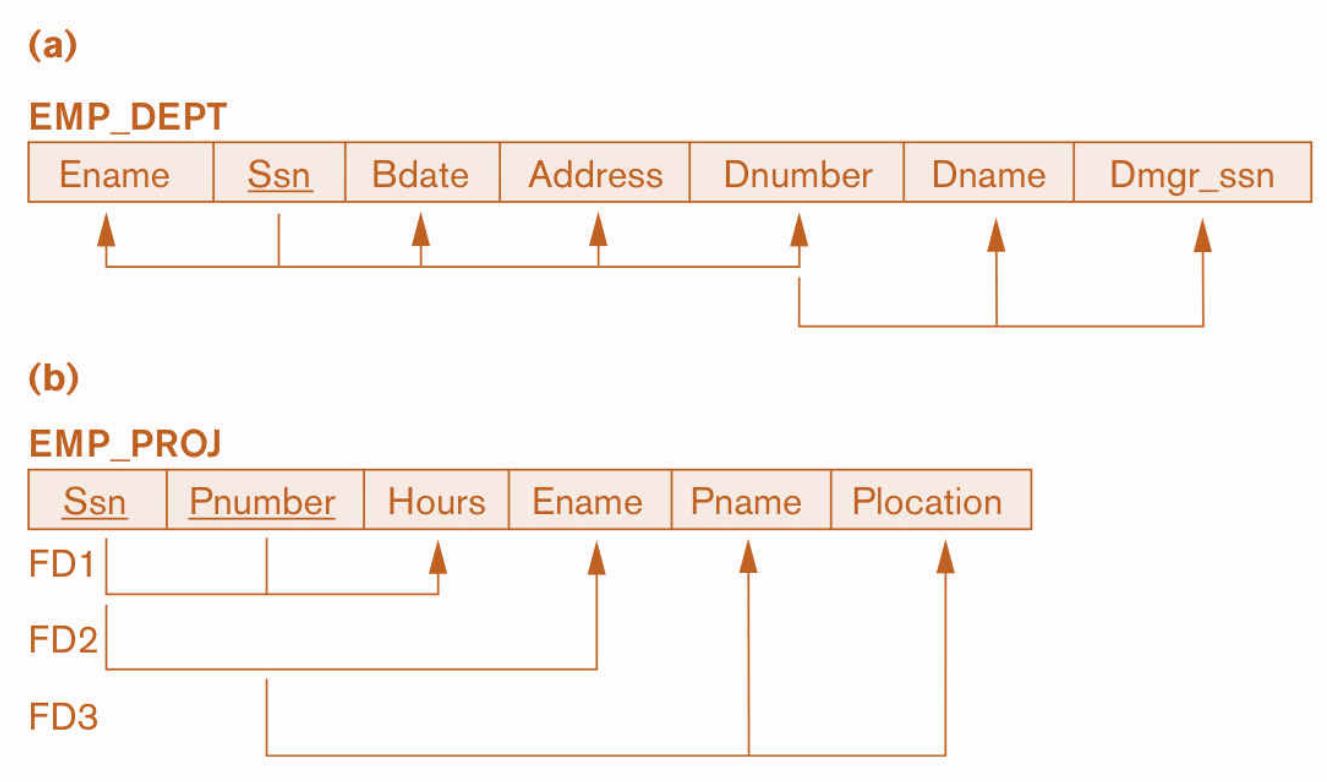


Figure 14.3
Two relation schemas suffering from update anomalies. (a) EMP_DEPT and (b) EMP_PROJ.

Database Management Systems

Sample states for EMP_DEPT and EMP_PROJ

Redundancy

EMP_DEPT

Ename	Ssn	Bdate	Address	Dnumber	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 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

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

Figure 14.4
Sample states for EMP_DEPT and EMP_PROJ resulting from applying NATURAL JOIN to the relations in Figure 14.2. These may be stored as base relations for performance reasons.

Guideline for 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 anomalies 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

NULL Values in Tuples

Many NULLs waste space at the storage level and may also lead to problems with understanding the meaning of the attributes

Guideline 3

As far as possible, avoid placing attributes in a base relation whose values may frequently be NULL

Example: if only 15 percent of employees have individual offices, there is little justification for including an attribute Office_number in the EMPLOYEE relation; rather, a relation EMP_OFFICES (Essn, Office_number) can be created to include tuples for only the employees with individual offices

Database Management Systems

Generation of Spurious Tuples – avoid at any cost

- 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.

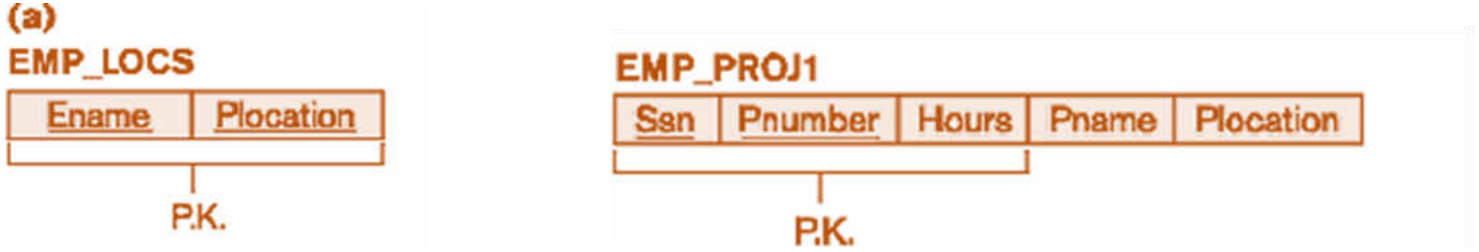
Generation of Spurious Tuples

- The two relations EMP_PROJ1 and EMP_LOCS as the base relations of EMP_PROJ, is not a good schema design.
- Problem is if a Natural Join is performed on the above two relations it produces more tuples than original set of tuples in EMP_PROJ.
- These additional tuples that were not in EMP_PROJ are called spurious tuples because they represent spurious or wrong information that is not valid.
- This is because the PLOCATION attribute which is used for joining is neither a primary key, nor a foreign key in either EMP_LOCS AND EMP_PROJ1.

Database Management Systems

Spurious Tuples – Example

(a)



(b)

EMP_LOCS

Ename	Plocation
Smith, John B.	Bellaire
Smith, John B.	Surgarland
Narayan, Ramesh K.	Houston
English, Joyce A.	Bellaire
English, Joyce A.	Surgarland
Wong, Franklin T.	Surgarland
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

Database Management Systems

Spurious Tuples – Example

Spurious tuples are represented by *

	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.

* *

Spurious Tuples (2)

- There are two important properties of decompositions:
  a) Non-additive or lossless 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.

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu

Database Management Systems

Functional Dependencies

Prof. Ruby Dinakar & Prof. Nivedita

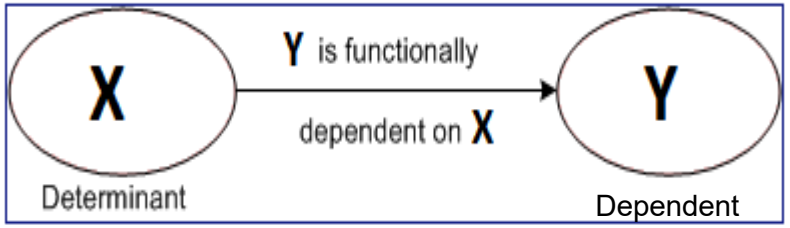
Department of Computer Science and Engineering

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the below resource and persons:
- Author slides from “Fundamentals of Database Systems”, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

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
- The functional dependency (FD) is a relationship that exists between two attributes, Where one attribute can directly or indirectly derived from the other attribute.
- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y



Functional dependency between X and Y

Database Management Systems

Defining Functional Dependencies

- 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)

- **The main use of FD is to describe further a relational schema R by specifying constraints on its attributes that must hold all the times.**
- 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])

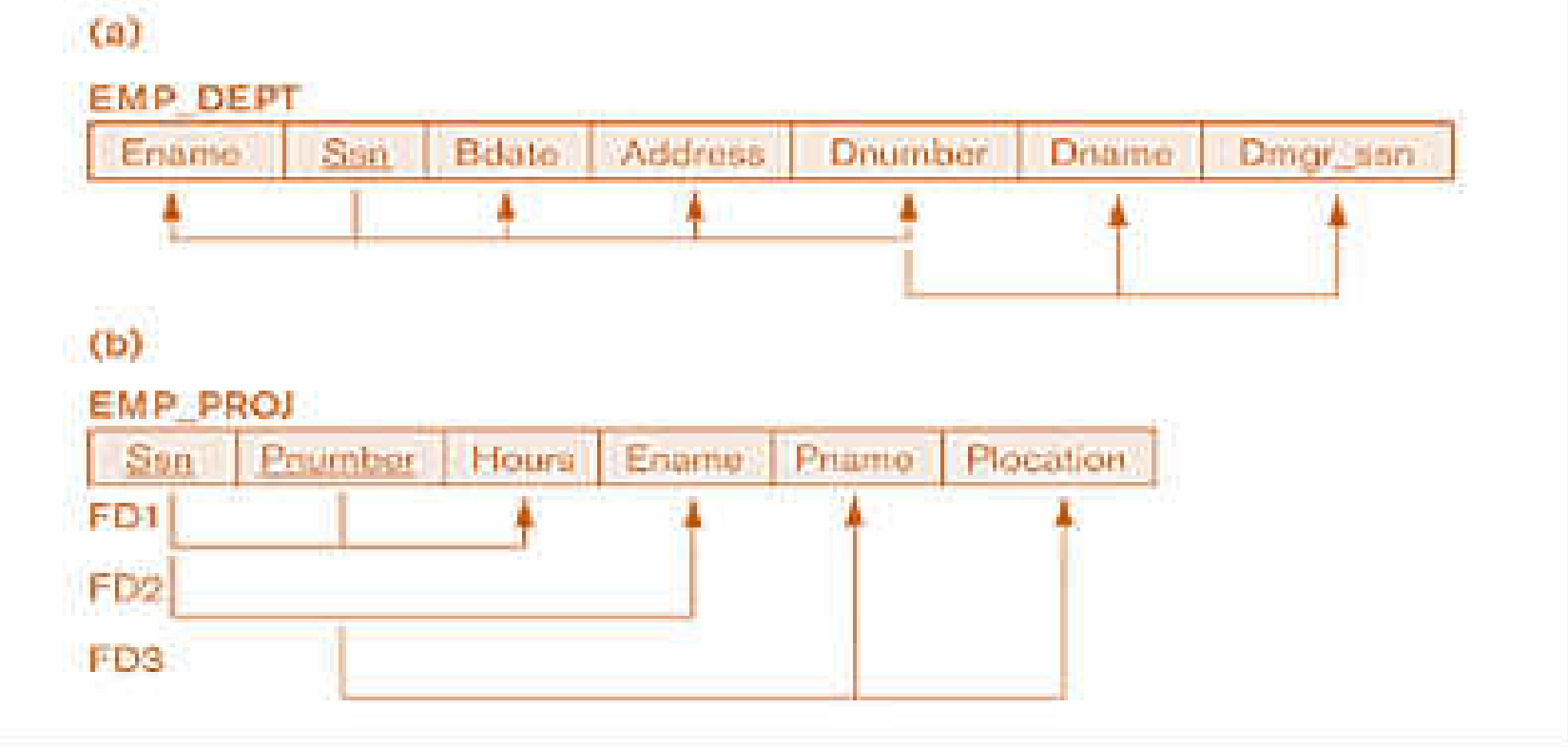
Defining FDs from instances

- Note that in order to define the FDs, *we need to understand the meaning of the attributes involved and the relationship between them.*
- An FD is a property of the attributes in the schema R
- Given the instance (population) of a relation, all we can conclude is that an FD *may exist* between certain attributes.
- What we can definitely conclude is – that certain FDs *do not exist* because there are tuples that show a violation of those dependencies.

Database Management Systems

Diagrammatic Notation for displaying FDs

- Each FD is displayed as a horizontal line.
- The left hand side attributes are connected by vertical lines to the line representing the FD, whereas the right hand side attributes are connected by the lines with arrows pointing toward the attributes.



Database Management Systems

Types of Functional Dependency

- **Full Functional Dependency:** A functional dependency is said to be full dependency "if and only if the determinant of the functional dependency if either candidate key or super key, and the dependent can be either prime or non-prime attribute".
- **Example:** Consider the following determinant ABC → D i.e., ABC determines D but D is not determined by any subset of A/ BC/C/B/AB i.e., BC→D, C→D, A→D Functional dependencies are not exists. So D is Fully Functional Dependent.
- **Partial Functional Dependency:** If a non-prime attribute of the relation is getting derived by only a part of the candidate key, then such dependency is known as Partial Dependency.
- **Example:** Consider the following determinants AC→P, A→D, D→P. From these determinants P is not fully FD on AC. Because, If we find A+ (means A's Closure) A→D, D→P i.e., A→P. But we don't have any requirement of C. C attribute is removed completely. So P is Partially Dependent on AC

Database Management Systems

Types of Functional Dependency

- **Transitive Functional Dependency:** If a non-prime attribute of a relation is getting derived by either another non-prime attribute or the combination of the part of the candidate key along with non-prime attribute, then such dependency is defined as Transitive dependency. i.e., in a relation, there may be dependency among non-key fields. Such dependency is called Transitive Functional Dependency.
- **Example:** X→Y, and Y→Z then we can determine X→Z holds.
- **Trivial Functional Dependency:** It is basically related to Reflexive rule. i.e., if X is a set of attributes, and Y is subset of X then X→Y holds.
- **Example:** ABC→BC is a Trivial Dependency.
- **Multi-Valued Dependency:** Consider 3 fields X, Y, and Z in a relation. If for each value of X, there is a well-defined set of values Y and Well-defined set of values of Z and set of values of Y is independent of the set values of Z. This dependency is Multi-valued Dependency. i.e., X →Y / Z.

Ruling Out FDs

Consider the following TEACH relation.
we can say that the FD: Text → Course may exist.
However, the FDs Teacher → Course, Teacher → Text and Couse → Text are ruled out.

TEACH

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

What FDs may exist?

- A relation *R*(A, B, C, D) with its extension.
- The following FDs may hold because the 4 tuples in the current extension have no violation of these constraints: B -> C; C->B; {A,B} -> C; {A,B} -> D; and {C,D} -> B
- The following do not hold: A-> B, B-> A ; D -> C

A	B	C	D
a1	b1	c1	d1
a1	b2	c2	d2
a2	b2	c2	d3
a3	b3	c4	d3

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu

Database Management Systems

Normal Forms

Prof. Ruby Dinakar & Prof. Nivedita

Department of Computer Science and Engineering

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the below resource and persons:
- Author slides from “Fundamentals of Database Systems”, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

14.1 Informal Design Guidelines for Relational Databases

14.2 Functional Dependencies (FDs)

14.3 Normal Forms Based on Primary Keys

14.4 General Normal Form Definitions (For Multiple Keys)

14. 5 BCNF (Boyce-Codd Normal Form)

14.6 Multivalued Dependency and Fourth Normal Form

14.7 Join Dependencies and Fifth Normal Form

Database Management Systems

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

Database Management Systems

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

Database Management Systems

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 and BCNF. 4NF rarely used in practice.)
- **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 = {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 (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

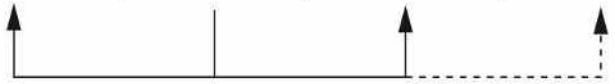
- Disallows
  - composite attributes
  - multivalued attributes
  - **nested relations**; attributes whose values for an *individual tuple* are non-atomic
- Considered to be part of the definition of a relation
- Most RDBMSs allow only those relations to be defined that are in First Normal Form

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 14.9
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Sample state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

Database Management Systems

Normalizing 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 14.10
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.

3.5 Second Normal Form (1)

- 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 -> 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

Database Management Systems

Second Normal Form (2)

- 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

- R can be decomposed into 2NF relations via the process of 2NF normalization or “second normalization”

Figure 14.11 Normalizing into 2NF and 3NF

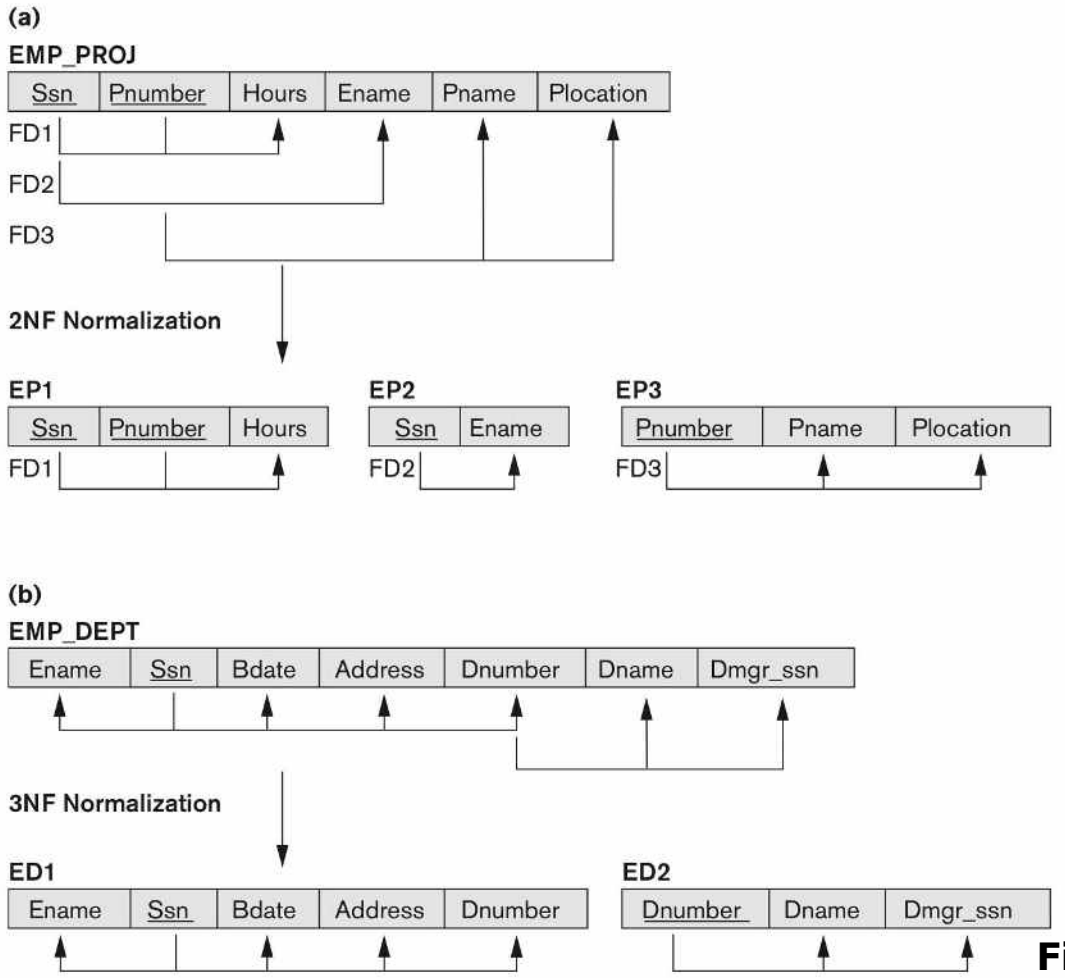


Figure 14.11
Normalizing into 2NF and 3NF. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

Database Management Systems

Figure 14.12 Normalization into 2NF and 3NF

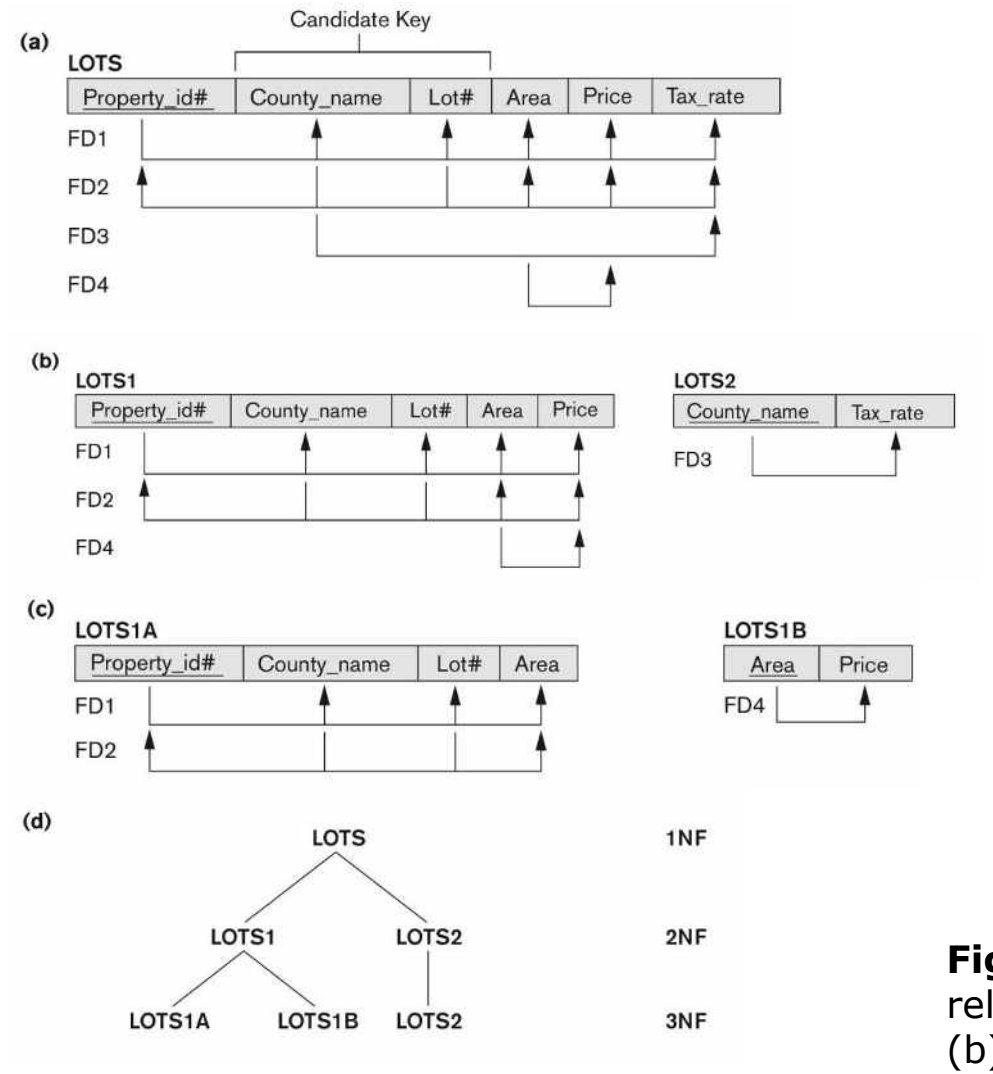


Figure 14.12 Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Progressive normalization of LOTS into a 3NF design.

3.6 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

3.6 Third Normal Form (2)

- 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
- 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 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.

Database Management Systems

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**

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
- Any attribute involved in a candidate key is a *prime attribute*
- All other attributes are called *non-prime attributes.*

General Definition of 2NF (For 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
- In Figure 14.12 the FD

County_name → Tax_rate violates 2NF.

So second normalization converts LOTS into

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

LOTS2 (County_name, Tax_rate)

General Definition of Third Normal Form

- 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
- LOTS1 relation violates 3NF because

Area → Price ; and Area is not a superkey in LOTS1. (see Figure 14.12).

Interpreting the General Definition of Third Normal Form

- Consider the 2 conditions in the Definition of 3NF:
- 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
- Condition (a) catches two types of violations :
- - one where a prime attribute functionally determines a non-prime attribute. This catches 2NF violations due to non-full functional dependencies.
- -second, where a non-prime attribute functionally determines a non-prime attribute. This catches 3NF violations due to a transitive dependency.

Interpreting the General Definition of Third Normal Form (2)

- **ALTERNATIVE DEFINITION of 3NF: We can restate the definition as:**

  A relation schema R is in **third normal form (3NF)** if every non-prime attribute in R meets both of these conditions:

  - It is fully functionally dependent on every key of R
  - It is non-transitively dependent on every key of R

  Note that stated this way, a relation in 3NF also meets the requirements for 2NF.

- The condition (b) from the last slide takes care of the dependencies that "slip through" (are allowable to) 3NF but are "caught by" BCNF which we discuss next.

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu

Database Management Systems

Normal Forms

Prof. Ruby Dinakar & Prof. Nivedita

Department of Computer Science and Engineering

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the below resource and persons:
- Author slides from “Fundamentals of Database Systems”, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

14.1 Informal Design Guidelines for Relational Databases

14.2 Functional Dependencies (FDs)

14.3 Normal Forms Based on Primary Keys

14.4 General Normal Form Definitions (For Multiple Keys)

14. 5 BCNF (Boyce-Codd Normal Form)

14.6 Multivalued Dependency and Fourth Normal Form

14.7 Join Dependencies and Fifth Normal Form

Database Management Systems

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

Database Management Systems

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

Database Management Systems

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 and BCNF. 4NF rarely used in practice.)
- **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 = {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 (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

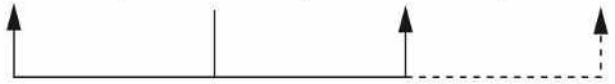
- Disallows
  - composite attributes
  - multivalued attributes
  - **nested relations**; attributes whose values for an *individual tuple* are non-atomic
- Considered to be part of the definition of a relation
- Most RDBMSs allow only those relations to be defined that are in First Normal Form

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 14.9
Normalization into 1NF. (a) A relation schema that is not in 1NF. (b) Sample state of relation DEPARTMENT. (c) 1NF version of the same relation with redundancy.

Database Management Systems

Normalizing 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 14.10
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.

3.5 Second Normal Form (1)

- 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 -> 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

Database Management Systems

Second Normal Form (2)

- 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

- R can be decomposed into 2NF relations via the process of 2NF normalization or “second normalization”

Figure 14.11 Normalizing into 2NF and 3NF

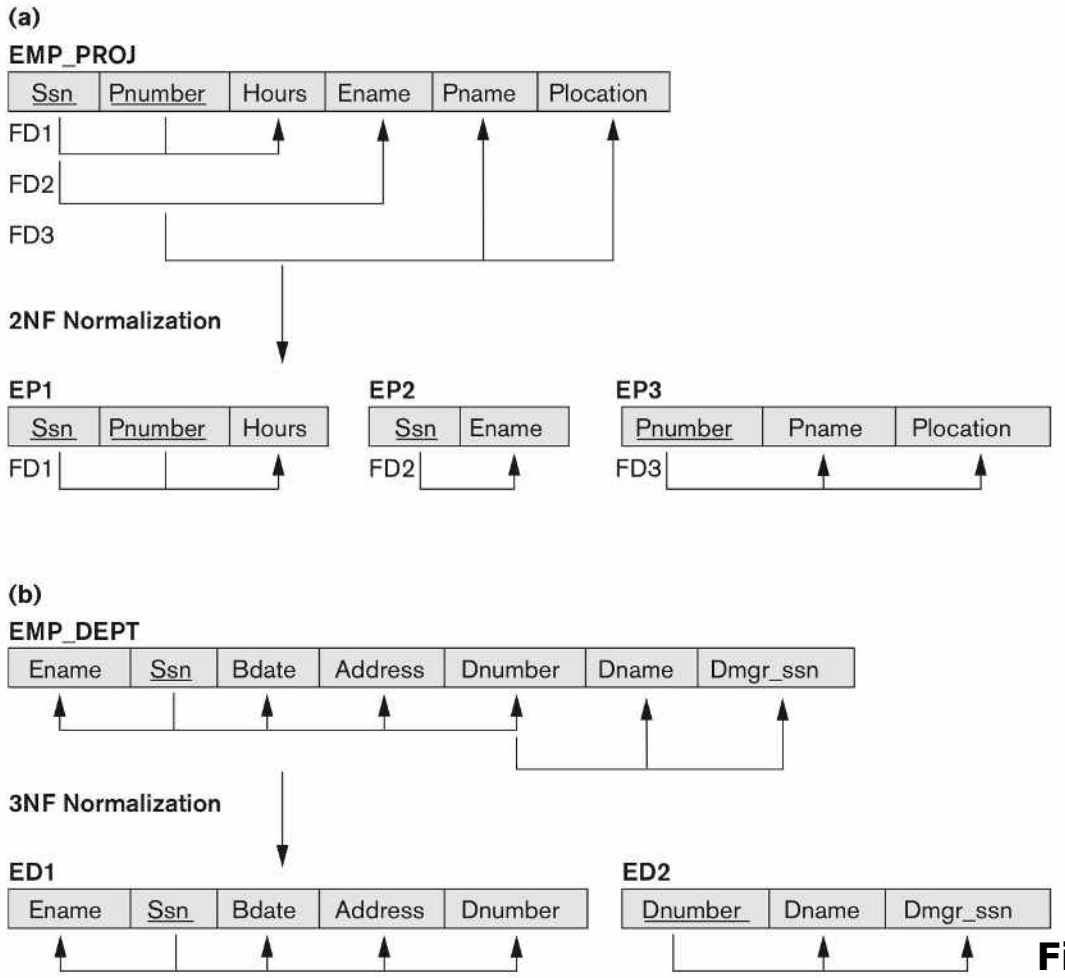


Figure 14.11
Normalizing into 2NF and 3NF. (a) Normalizing EMP_PROJ into 2NF relations. (b) Normalizing EMP_DEPT into 3NF relations.

Database Management Systems

Figure 14.12 Normalization into 2NF and 3NF

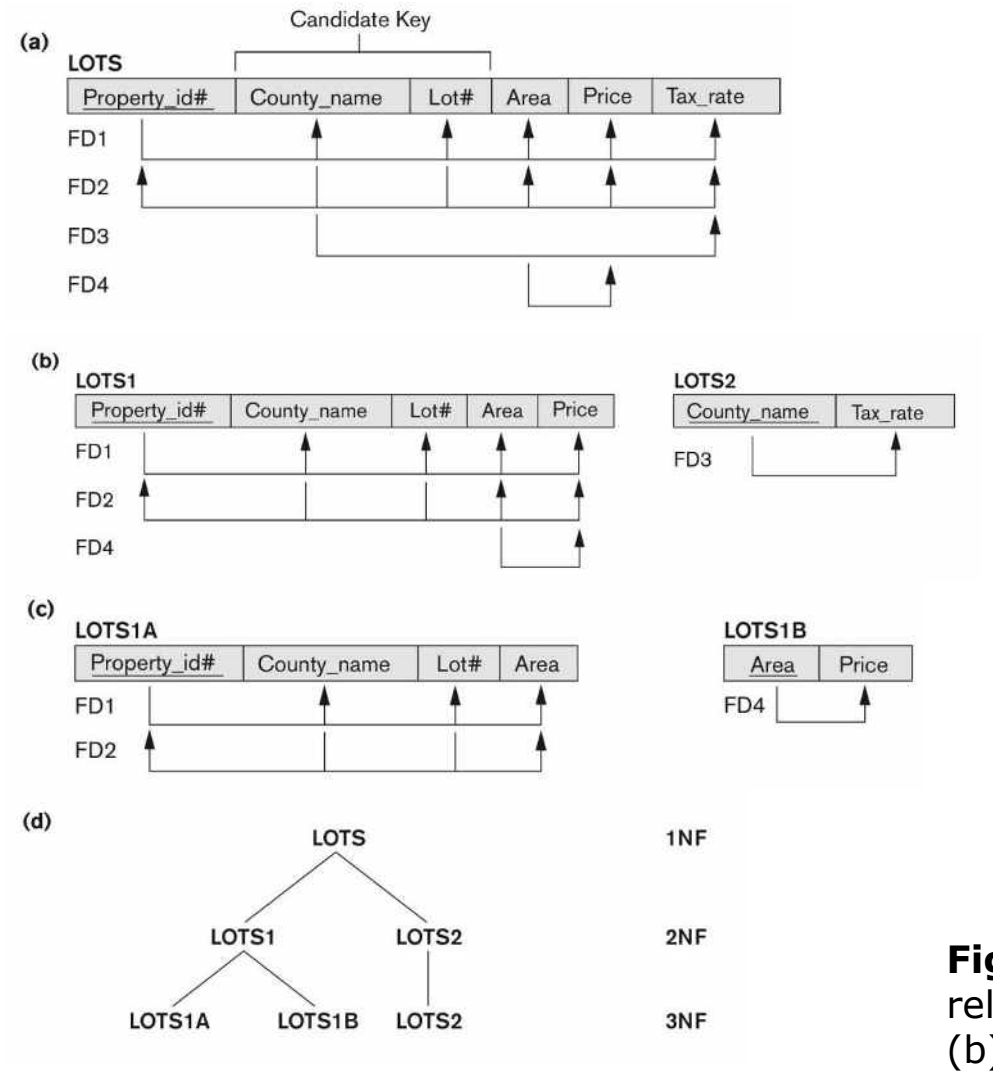


Figure 14.12 Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Progressive normalization of LOTS into a 3NF design.

3.6 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

3.6 Third Normal Form (2)

- 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
- 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 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.

Database Management Systems

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**

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
- Any attribute involved in a candidate key is a *prime attribute*
- All other attributes are called *non-prime attributes.*

General Definition of 2NF (For 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
- In Figure 14.12 the FD

County_name → Tax_rate violates 2NF.

So second normalization converts LOTS into

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

LOTS2 (County_name, Tax_rate)

General Definition of Third Normal Form

- 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
- LOTS1 relation violates 3NF because

Area → Price ; and Area is not a superkey in LOTS1. (see Figure 14.12).

Interpreting the General Definition of Third Normal Form

- Consider the 2 conditions in the Definition of 3NF:
- 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
- Condition (a) catches two types of violations :
- - one where a prime attribute functionally determines a non-prime attribute. This catches 2NF violations due to non-full functional dependencies.
- -second, where a non-prime attribute functionally determines a non-prime attribute. This catches 3NF violations due to a transitive dependency.

Interpreting the General Definition of Third Normal Form (2)

- **ALTERNATIVE DEFINITION of 3NF: We can restate the definition as:**

  A relation schema R is in **third normal form (3NF)** if every non-prime attribute in R meets both of these conditions:

  - It is fully functionally dependent on every key of R
  - It is non-transitively dependent on every key of R

  Note that stated this way, a relation in 3NF also meets the requirements for 2NF.

- The condition (b) from the last slide takes care of the dependencies that "slip through" (are allowable to) 3NF but are "caught by" BCNF which we discuss next.

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu

Database Management Systems

BCNF and overview of Higher Normal Forms

Prof. Ruby Dinakar & Prof. Nivedita

Department of Computer Science and Engineering

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the below resource and persons:
- Author slides from “Fundamentals of Database Systems”, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

14.1 Informal Design Guidelines for Relational Databases

14.2 Functional Dependencies (FDs)

14.3 Normal Forms Based on Primary Keys

14.4 General Normal Form Definitions (For Multiple Keys)

14. 5 BCNF (Boyce-Codd Normal Form)

14.6 Multivalued Dependency and Fourth Normal Form

14.7 Join Dependencies and Fifth Normal Form

BCNF (Boyce-Codd Normal Form)

- A relation schema R is in Boyce-Codd Normal Form (BCNF) if whenever an FD $X \rightarrow A$ 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
- Hence BCNF is considered a stronger form of 3NF
- The goal is to have each relation in BCNF (or 3NF)

Fig. 14.13 BCNF (Boyce-Codd Normal Form)

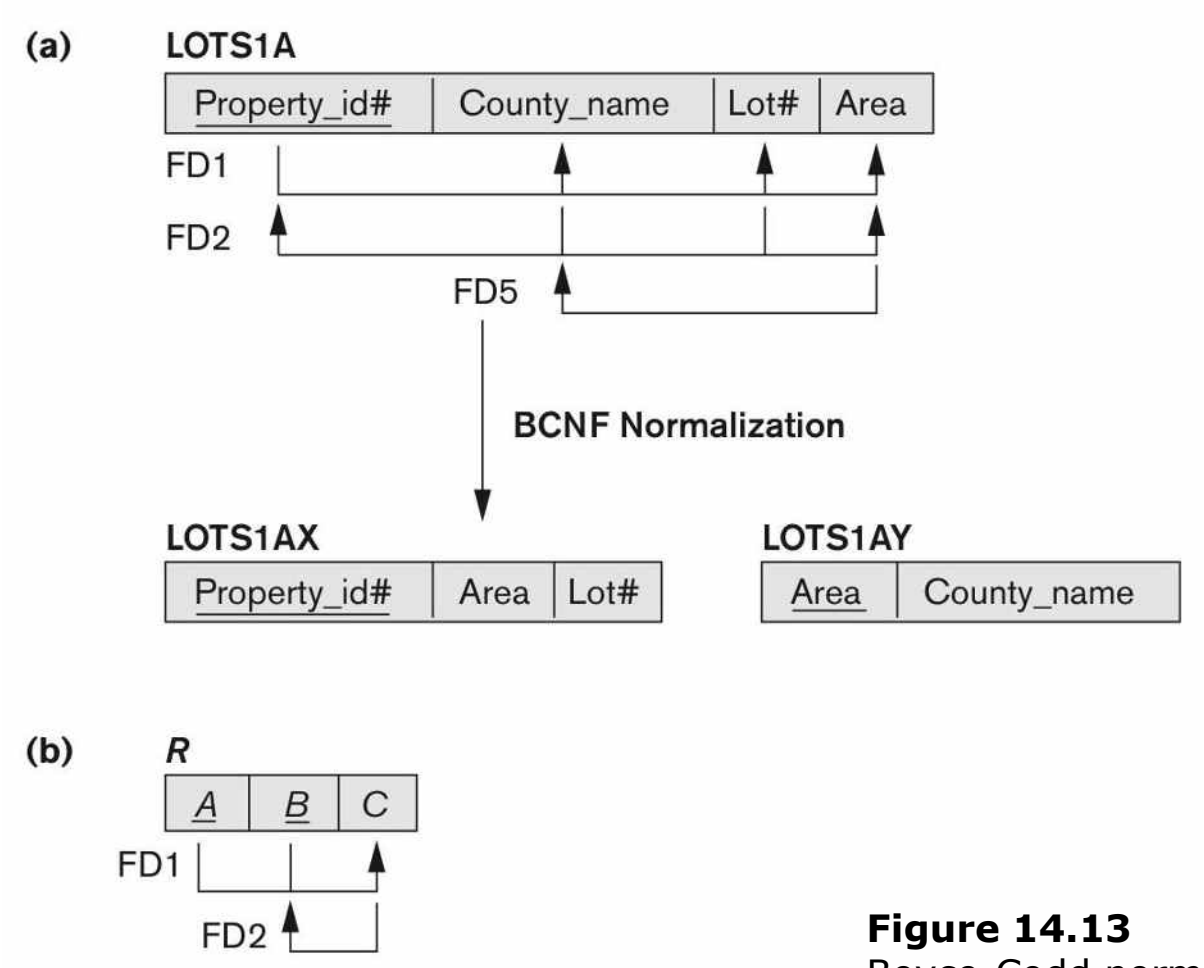


Figure 14.13
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF due to the f.d. C → B.

Figure 14.14 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
Narayan	Operating Systems	Ammar

Figure 14.14
A relation TEACH that is in 3NF but not BCNF.

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 14.13 (b).
  - So this relation is in 3NF *but not in* 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 15.3)

Achieving the BCNF by Decomposition (2)

- Three possible decompositions for relation TEACH
  - D1: {student, instructor} and {student, course}
  - D2: {course, instructor } and {course, student}
  - D3: {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 non-additive (lossless) is discussed under Property NJB on the next slide. We then show how the third decomposition above meets the property.

Test for checking non-additivity of Binary Relational Decompositions

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

Test for checking non-additivity of Binary Relational Decompositions

If you apply the NJB test to the 3 decompositions of the TEACH relation:

- D1 gives **Student** → Instructor or **Student** → Course, none of which is true.
- D2 gives **Course** → Instructor or **Course** → Student, none of which is true.
- However, in D3 we get **Instructor** → Course or **Instructor** → Student.

Since **Instructor** → Course is indeed true, the NJB property is satisfied and D3 is determined as a non-additive (good) decomposition.

General Procedure for achieving BCNF when a relation fails BCNF

- Let R be the relation not in BCNF, let X be a subset-of R, and let $X \rightarrow A$ be the FD that causes a violation of BCNF. Then R may be decomposed into two relations:
- (i) $R - A$ and (ii) $X \cup A$.
- If either $R - A$ or X u A. is not in BCNF, repeat the process.

Note that the f.d. that violated BCNF in TEACH was Instructor $\rightarrow$Course. Hence its BCNF decomposition would be :

(TEACH – COURSE) and (Instructor u Course), which gives

the relations: (Instructor, Student) and (Instructor, Course) that we obtained before in decomposition D3.

Important points for BCNF

•Every binary relation (a relation with only two attributes) is always in BCNF.
•BCNF is free from redundancies arising out of functional dependencies (zero redundancy).
•A relation with only trivial functional dependencies is always in BCNF. In other words, a relation with no non-trivial functional dependencies is always in BCNF.
•BCNF decomposition is always lossless but not always dependency preserving.

Sometimes, going for BCNF may not preserve functional dependencies. So, go for BCNF only if the lost functional dependencies are not required else normalize till 3NF only.

•There exist many more normal forms even after BCNF like 4NF and more.
•But in the real world database systems, it is generally not required to go beyond BCNF.

Multivalued Dependencies and Fourth Normal Form (1)

Definition:

- A **multivalued dependency** (**MVD**) X —>> 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 —>> Y in R is called a **trivial MVD** if (a) Y is a subset of X, or (b) $X \cup Y = R$.

Database Management Systems

Multivalued Dependencies and Fourth Normal Form (3)

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.

We can state the following points:

- An all-key relation is always in BCNF since it has no FDs.
- An all key relation such as the EMP relation in fig 14.15(a), which has no FDs but has MVD Ename ->> Pname | Dname, is not in 4NF.
- A relation that is not in 4NF due to non trivial MVD must be decomposed to convert it into a set of relations in 4NF.
- The decomposition removes the redundancy caused by the MVD.

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*.

Database Management Systems

Fourth and fifth normal forms.

(a) EMP

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

(b) EMP_PROJECTS

Ename	Pname
Smith	X
Smith	Y

EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

(c) SUPPLY

Sname	Part_name	Proj_name
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

(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

Figure 14.15 Fourth and fifth normal forms. (a) The EMP relation with two MVDs: Ename –>> Pname and Ename –>> Dname. (b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS. (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, R3.

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)

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.

- Discovering join dependencies in practical databases with hundreds of relations is next to impossible. Therefore, 5NF is rarely used in practice.

Database Management Systems

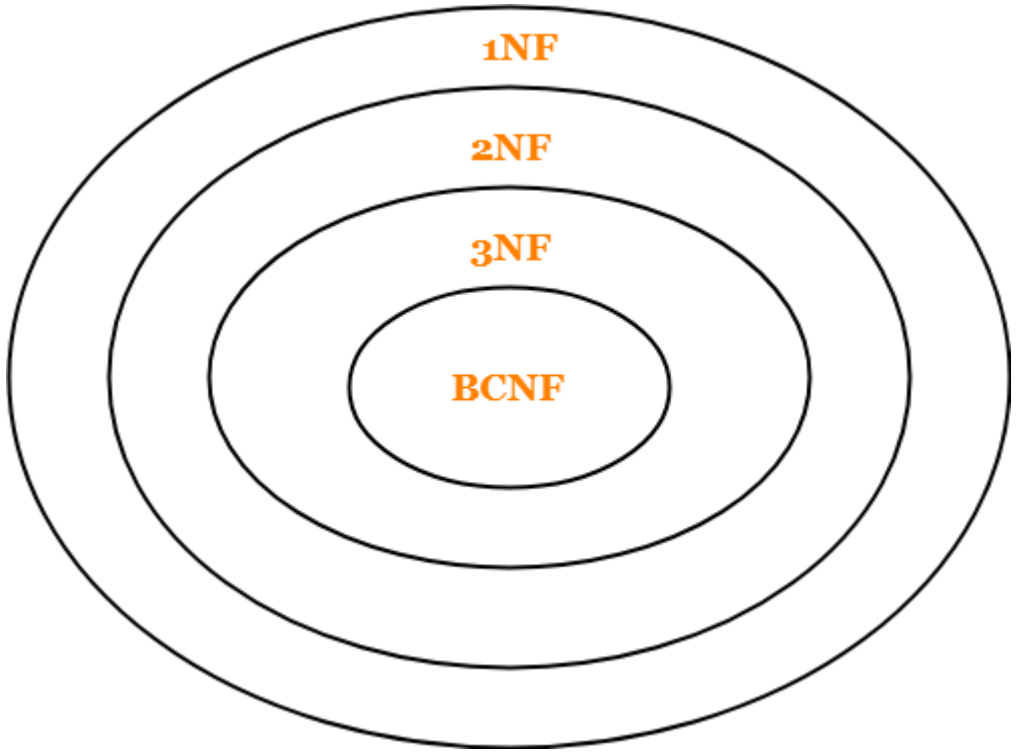
Chapter Summary

- Informal Design Guidelines for Relational Databases
- Functional Dependencies (FDs)
- Normal Forms (1NF, 2NF, 3NF)Based on Primary Keys
- General Normal Form Definitions of 2NF and 3NF (For Multiple Keys)
- BCNF (Boyce-Codd Normal Form)
- Fourth and Fifth Normal Forms

Chapter Summary

Remember the following diagram which implies-

- A relation in BCNF will surely be in all other normal forms.
- A relation in 3NF will surely be in 2NF and 1NF.
- A relation in 2NF will surely be in 1NF.



This diagram also implies-

- BCNF is stricter than 3NF.
- 3NF is stricter than 2NF.
- 2NF is stricter than 1NF.

Chapter Summary

While determining the normal form of any given relation,

•Start checking from BCNF.

•This is because if it is found to be in BCNF, then it will surely be in all other normal forms.

•If the relation is not in BCNF, then start moving towards the outer circles and check for other normal forms in the order they appear.

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu

Database Management Systems

BCNF and overview of Higher Normal Forms

Prof. Ruby Dinakar & Prof. Nivedita

Department of Computer Science and Engineering

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the below resource and persons:
- Author slides from “Fundamentals of Database Systems”, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

14.1 Informal Design Guidelines for Relational Databases

14.2 Functional Dependencies (FDs)

14.3 Normal Forms Based on Primary Keys

14.4 General Normal Form Definitions (For Multiple Keys)

14. 5 BCNF (Boyce-Codd Normal Form)

14.6 Multivalued Dependency and Fourth Normal Form

14.7 Join Dependencies and Fifth Normal Form

BCNF (Boyce-Codd Normal Form)

- A relation schema R is in Boyce-Codd Normal Form (BCNF) if whenever an FD $X \rightarrow A$ 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
- Hence BCNF is considered a stronger form of 3NF
- The goal is to have each relation in BCNF (or 3NF)

Fig. 14.13 BCNF (Boyce-Codd Normal Form)

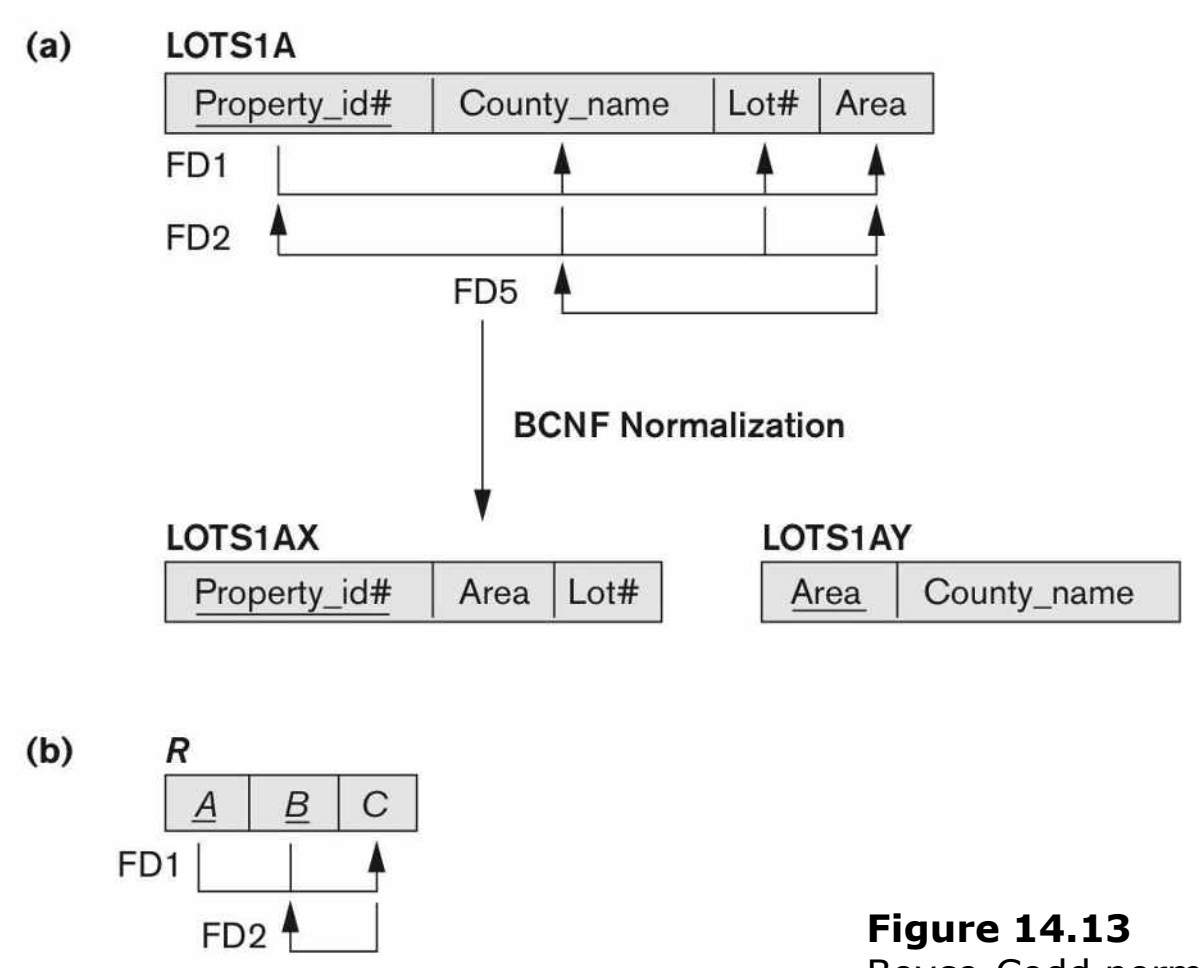


Figure 14.13
Boyce-Codd normal form. (a) BCNF normalization of LOTS1A with the functional dependency FD2 being lost in the decomposition. (b) A schematic relation with FDs; it is in 3NF, but not in BCNF due to the f.d. C → B.

Figure 14.14 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
Narayan	Operating Systems	Ammar

Figure 14.14
A relation TEACH that is in 3NF but not BCNF.

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 14.13 (b).
  - So this relation is in 3NF *but not in* 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 15.3)

Achieving the BCNF by Decomposition (2)

- Three possible decompositions for relation TEACH
  - D1: {student, instructor} and {student, course}
  - D2: {course, instructor } and {course, student}
  - D3: {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 non-additive (lossless) is discussed under Property NJB on the next slide. We then show how the third decomposition above meets the property.

Test for checking non-additivity of Binary Relational Decompositions

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

Test for checking non-additivity of Binary Relational Decompositions

If you apply the NJB test to the 3 decompositions of the TEACH relation:

- D1 gives **Student** → Instructor or **Student** → Course, none of which is true.
- D2 gives **Course** → Instructor or **Course** → Student, none of which is true.
- However, in D3 we get **Instructor** → Course or **Instructor** → Student.

Since **Instructor** → Course is indeed true, the NJB property is satisfied and D3 is determined as a non-additive (good) decomposition.

General Procedure for achieving BCNF when a relation fails BCNF

- Let R be the relation not in BCNF, let X be a subset-of R, and let $X \rightarrow A$ be the FD that causes a violation of BCNF. Then R may be decomposed into two relations:
- (i) $R - A$ and (ii) $X \cup A$.
- If either $R - A$ or X u A. is not in BCNF, repeat the process.

Note that the f.d. that violated BCNF in TEACH was Instructor $\rightarrow$Course. Hence its BCNF decomposition would be :

(TEACH – COURSE) and (Instructor u Course), which gives

the relations: (Instructor, Student) and (Instructor, Course) that we obtained before in decomposition D3.

Important points for BCNF

•Every binary relation (a relation with only two attributes) is always in BCNF.
•BCNF is free from redundancies arising out of functional dependencies (zero redundancy).
•A relation with only trivial functional dependencies is always in BCNF. In other words, a relation with no non-trivial functional dependencies is always in BCNF.
•BCNF decomposition is always lossless but not always dependency preserving.

Sometimes, going for BCNF may not preserve functional dependencies. So, go for BCNF only if the lost functional dependencies are not required else normalize till 3NF only.

•There exist many more normal forms even after BCNF like 4NF and more.
•But in the real world database systems, it is generally not required to go beyond BCNF.

Multivalued Dependencies and Fourth Normal Form (1)

Definition:

- A **multivalued dependency** (**MVD**) X —>> 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 —>> Y in R is called a **trivial MVD** if (a) Y is a subset of X, or (b) $X \cup Y = R$.

Database Management Systems

Multivalued Dependencies and Fourth Normal Form (3)

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.

We can state the following points:

- An all-key relation is always in BCNF since it has no FDs.
- An all key relation such as the EMP relation in fig 14.15(a), which has no FDs but has MVD Ename ->> Pname | Dname, is not in 4NF.
- A relation that is not in 4NF due to non trivial MVD must be decomposed to convert it into a set of relations in 4NF.
- The decomposition removes the redundancy caused by the MVD.

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*.

Database Management Systems

Fourth and fifth normal forms.

(a) EMP

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

(b) EMP_PROJECTS

Ename	Pname
Smith	X
Smith	Y

EMP_DEPENDENTS

Ename	Dname
Smith	John
Smith	Anna

(c) SUPPLY

Sname	Part_name	Proj_name
Smith	Bolt	ProjX
Smith	Nut	ProjY
Adamsky	Bolt	ProjY
Walton	Nut	ProjZ
Adamsky	Nail	ProjX
Adamsky	Bolt	ProjX
Smith	Bolt	ProjY

(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

Figure 14.15 Fourth and fifth normal forms. (a) The EMP relation with two MVDs: Ename –>> Pname and Ename –>> Dname. (b) Decomposing the EMP relation into two 4NF relations EMP_PROJECTS and EMP_DEPENDENTS. (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, R3.

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)

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.

- Discovering join dependencies in practical databases with hundreds of relations is next to impossible. Therefore, 5NF is rarely used in practice.

Database Management Systems

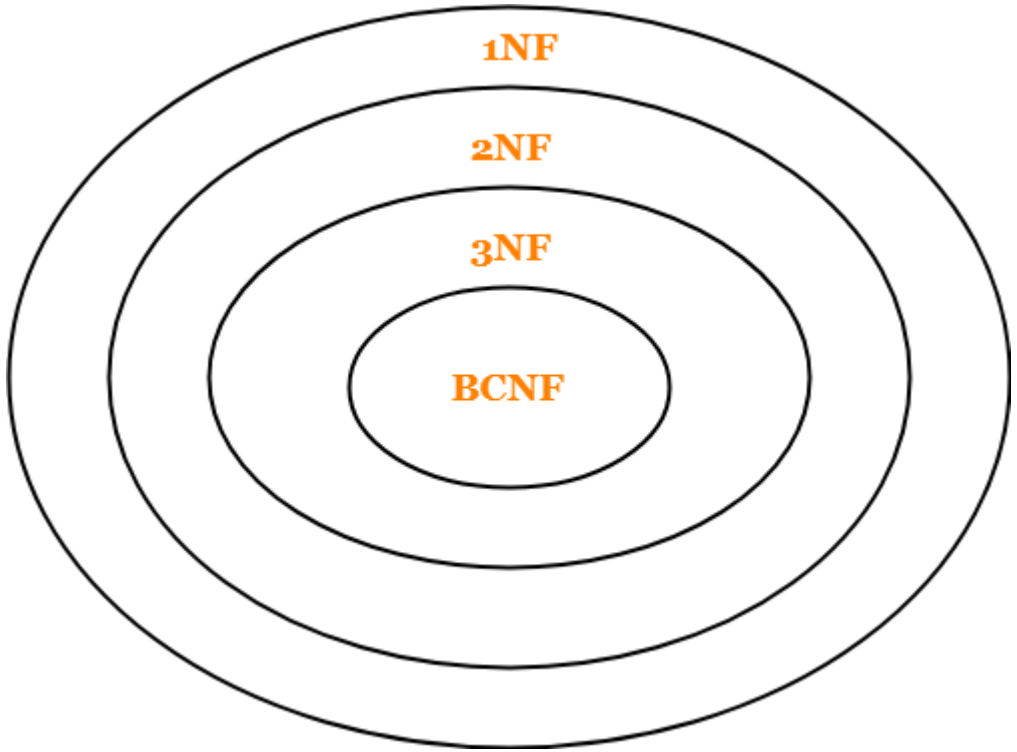
Chapter Summary

- Informal Design Guidelines for Relational Databases
- Functional Dependencies (FDs)
- Normal Forms (1NF, 2NF, 3NF)Based on Primary Keys
- General Normal Form Definitions of 2NF and 3NF (For Multiple Keys)
- BCNF (Boyce-Codd Normal Form)
- Fourth and Fifth Normal Forms

Chapter Summary

Remember the following diagram which implies-

- A relation in BCNF will surely be in all other normal forms.
- A relation in 3NF will surely be in 2NF and 1NF.
- A relation in 2NF will surely be in 1NF.



This diagram also implies-

- BCNF is stricter than 3NF.
- 3NF is stricter than 2NF.
- 2NF is stricter than 1NF.

Chapter Summary

While determining the normal form of any given relation,

•Start checking from BCNF.

•This is because if it is found to be in BCNF, then it will surely be in all other normal forms.

•If the relation is not in BCNF, then start moving towards the outer circles and check for other normal forms in the order they appear.

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu

Database Management Systems

Inference Rules

Prof. Ruby Dinakar & Prof. Nivedita

Department of Computer Science and Engineering

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the below resource and persons:
- Author slides from “Fundamentals of Database Systems”, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

15.1 Further topics in Functional Dependencies: Inference Rules, Equivalence and Minimal Cover

15.2 Properties of Relational Decomposition

15.3 Algorithms for Relational Database Schema Design

15.4 About Nulls, Dangling Tuples and Alternative Relational Designs

Functional Dependencies : Inference Rules, Equivalence and Minimal Cover

- A set of attributes X functionally determines a set of attributes Y if the value of X determines a unique value for Y.

- Our goal here is to determine the properties of functional dependencies and to find out the ways of manipulating them.

Defining Functional Dependencies

- $X \rightarrow Y$ 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]
- $X \rightarrow Y$ in R specifies a *constraint* on all relation instances r(R)
- FDs are derived from the real-world constraints on the attributes

Inference Rules for FDs (1)

- **Definition:** An FD $X \rightarrow Y$ is **inferred from** or **implied by** a set of dependencies F specified on R if $X \rightarrow Y$ holds in *every* legal relation state r of R; that is, whenever r satisfies all the dependencies in F, $X \rightarrow Y$ also holds in r.

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

Inference Rules for FDs (2)

- Armstrong's inference rules:
  - IR1. (**Reflexive**) If Y is a *subset-of* X, then X → Y always holds.
  - IR2. (**Augmentation**) If X → Y, then XZ → YZ always holds.
    - (Notation: XZ stands for X U Z)
    - Example: if {Ssn} → {Ename} then {Ssn, Bdate}→ {Ename, Bdate}
  - IR3. (**Transitive**) If X → Y and Y → Z, then X → Z always holds.
    - Example: if {Ssn} → {Dnumber} and {Dnumber} → {Dname} then {Ssn}→ {Dname}
- IR1, IR2, IR3 form a **sound** and **complete** set of inference rules
  - These are rules hold and all other rules that hold can be deduced from these

Inference Rules for FDs (3)

- Some additional inference rules that are useful:
  - Decomposition: If X → YZ, then X → Y and X → Z
  - Union/Additive: If X → Y and X → Z, then X → YZ
  - Example: if {Ssn} → {Ename} and {Ssn} → {Address} then {Ssn}→ {Ename, Address}
  - 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)

Example 1

Supposing we are given a relation R{A, B, C, D, E, F} with a set of FDs as shown below:

$A \rightarrow BC$

$B \rightarrow E$

$CD \rightarrow EF$

Let us show that the FD $AD \rightarrow F$ holds for R and is a member of the closure.

(1) $A \rightarrow BC$ {Given}

(2) $A \rightarrow C$ {Decomposition of (1)}

(3) $AD \rightarrow CD$ {Augmentation of (2) by adding D}

(4) $CD \rightarrow EF$ {Given}

(5) $AD \rightarrow EF$ {Transitivity of (3) and (4)}

(6) $AD \rightarrow F$ {Decomposition of (5)}

Database Management Systems

Example 2

Consider R = (Street, Zip, City) ; and the dependencies F = { City Street -> Zip, Zip -> City }

Show that: Street Zip ->Street Zip City

1. Zip ->City – Given
2. Street Zip -> Street City – Augmentation of (1) by Street
3. City Street -> Zip – Given
4. City Street -> City Street Zip – Augmentation of (3) by City Street
5. Street Zip -> City Street Zip – Transitivity of (2) and (4)

Database Management Systems

Example 3

Consider the relation schema <R,F> where R= (ABCDEGHI) and dependencies

F= { AB->E,AG->J, BE->I, E ->G GI ->H }. Show that AB ->GH is derived by F.

Step	Statement	Explanation
1	AB ->E	Given
2	E ->G	Given
3	BE->I	Given
4	GI ->H	Given
5	AB ->G	Transitivity on (1) and (2)
6	AB -> BE	Augmentation (1) by B
7	AB -> I	Transitivity on (6) and (3)
8	AB -> GI	Union on (5) and (7)
9	AB -> H	Transitivity on (8) and (4)
10	AB -> GH	Union on (5) and (9)

Closure

- Closure of a set F of FDs is the set F+ of all FDs that can be inferred from 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

Algorithm to determine Closure

- Algorithm 15.1. 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 → Z in F do
            if X+ ⊇ Y then X+ := X+ ∪ Z;
until (X+ = oldX+);
```

Database Management Systems

Example of Closure

- For example, consider the following relation schema about classes held at a university in a given academic year.

  CLASS (Classid, Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity).

- Let *F*, the set of functional dependencies for the above relation include the following FD's.

FD1: Classid → Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity;
FD2: Course# → Credit_hrs;
FD3: {Course#, Instr_name} → Text, Classroom;
FD4: Text → Publisher
FD5: Classroom → Capacity

These FD's. above represent the meaning of the individual attributes and the relationship among them and defines certain rules about the classes.

Example of Closure (cont.)

- The closures of attributes or sets of attributes for some example sets:

{ Classid } $^{+}$ = { Classid , Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity } = CLASS

{ Course#} $^{+}$ = { Course#, Credit_hrs}

{ Course#, Instr_name } $^{+}$ = { Course#, Credit_hrs, Text, Publisher, Classroom, Capacity }

Note that each closure above has an interpretation that is revealing about the attribute(s) on the left-hand-side. The closure of { Classid } $^{+}$ is the entire relation CLASS indicating that all attributes of the relation can be determined from Classid and hence it is a key.

Example of Closure (cont.)

- The closures of attributes or sets of attributes for some example sets:

{ Classid } $^{+}$ = { Classid , Course#, Instr_name, Credit_hrs, Text, Publisher, Classroom, Capacity } = CLASS

{ Course#} $^{+}$ = { Course#, Credit_hrs}

{ Course#, Instr_name } $^{+}$ = { Course#, Credit_hrs, Text, Publisher, Classroom, Capacity }

Note that each closure above has an interpretation that is revealing about the attribute(s) on the left-hand-side. The closure of { Classid } $^{+}$ is the entire relation CLASS indicating that all attributes of the relation can be determined from Classid and hence it is a key.

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu

Database Management Systems

Equivalence and Minimal Cover

Prof. Ruby Dinakar & Prof. Nivedita

Department of Computer Science and Engineering

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the below resource and persons:
- Author slides from “Fundamentals of Database Systems”, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

15.1 Further topics in Functional Dependencies: Inference Rules, Equivalence and Minimal Cover

15.2 Properties of Relational Decomposition

15.3 Algorithms for Relational Database Schema Design

15.4 About Nulls, Dangling Tuples and Alternative Relational Designs

Equivalence of Sets of FDs

- Two sets of FDs F and G are **equivalent** if:
  - Every FD in F can be inferred from G (G covers F), and
  - Every FD in G can be inferred from F (F covers G),
  - 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., F ⊇ G)

- F and G are equivalent if F covers G and G covers F

- There is an algorithm for checking equivalence of sets of FDs

Finding Minimal Cover of F.D.s

- Just as we applied inference rules to expand on a set *F* of FDs to arrive at *F*+, its closure, it is possible to think in the opposite direction to see if we could shrink or reduce the set *F* to its *minimal form* so that the minimal set is still equivalent to the original set *F.*

- Will use the concept of an **extraneous attribute** in a FD for defining the minimum cover.

- **Definition:** An attribute in a functional dependency is considered **extraneous attribute** if we can remove it without changing the closure of the set of dependencies. Formally, given F, the set of functional dependencies and a functional dependency $X \rightarrow A$ in F , attribute Y is extraneous in X if *Y is a subset of X,* and F logically implies $(F - (X \rightarrow A) \cup \{ (X - Y) \rightarrow A \})$

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 RHS.

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. (removing the redundancies via extraneous attributes on the LHS of a dependency)

3. We cannot remove any dependency from F and still have a set of dependencies that is equivalent to F. (removing the redundancies via having a dependency that can be inferred from the remaining FD s in F)

Minimal Sets of FDs

- **DEFINITION: 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 15.2. 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 $X \rightarrow \{A1, A2, ..., An\}$ in F by the n functional dependencies $X \rightarrow A1$, $X \rightarrow A2$, ..., $X \rightarrow An$. **(*Place FDs in a canonical form, Preparatory step*)**

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.
   (* The above constitutes a removal of the extraneous attribute B from X *)

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.
   (* The above constitutes a removal of the redundant dependency $X \rightarrow A$ from F *)

Computing the Minimal Sets of FDs: Example 1

Let the given set of FDs be *E* : {*B* → *A*, *D* → *A*, *AB* → *D*}.
Find the minimum cover of *E*.

- **Step 1:** All above dependencies are in **canonical** (they have only one attribute on the RHS) ;

Computing the Minimal Sets of FDs: Example 1

- **In step 2:** we need to determine if $AB \rightarrow D$ has any redundant **(EXTRANEOUS) attribute** on the left-hand side; that is, can it be replaced by $B \rightarrow D$ or $A \rightarrow D$?
- 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. Hence 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.

Computing the Minimal Sets of FDs: Example 1

- In **step 3** we look for a **redundant FD in E'.** By using the transitive rule on $B \rightarrow D$ and $D \rightarrow A$, we derive $B \rightarrow A$. Hence $B \rightarrow A$ is redundant in E' and can be eliminated.
- Hence the minimum cover of E is F: $\{B \rightarrow D, D \rightarrow A\}$.

The original set F can be inferred from E; In other words, the two sets F and E are equivalent.

Computing the Minimal Sets of FDs: Example 2

- **Step 1:** All above dependencies are not in **canonical** (they do not have one attribute on the RHS) ; so we have to convert them into:

F: { A -> B, A -> C, A -> D, A -> E, CD -> E}

Computing the Minimal Sets of FDs: Example 2

Let the given set of FDs be G : {A → BCDE, *CD* → E}.We have to find the minimum cover of G.

- **Step 1 :** F: { A -> B, A -> C, A -> D, A -> E, CD -> E}

- **Step 2 :** For CD - > E, neither C nor D is extra on the LHS, since we can not show that C -> E OR D -> E from the given FDs. Hence we can not replace it with either.

Computing the Minimal Sets of FDs: Example 2

- **Step 1 :** F: { A -> B, A -> C, A -> D, A -> E, CD -> E}
- **Step 2 :** For CD - > E, neither C nor D is extra on the LHS, since we can not show that C -> E OR D -> E from the given FDs. Hence we can not replace it with either.
- **Step 3 :** check any FD is redundant? Since A -> CD and CD -> E, by transitive rule (IR3), we get A -> E, Thus A -> E is redundant in G

Hence Minimum Cover of G, F : { A -> BCD, CD -> E}

Algorithm to determine the key of a relation

- **Algorithm 15.2a Finding a Key K for R, given a set F of Functional Dependencies**
  - **Input: A universal relation R and a set of functional dependencies F on the attributes of R.**

1. Set K := R;

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} };

In algorithm, we start by setting K to all the attributes of R; we can say that R itself is always a default super key. We then remove one attribute at a time and check whether the remaining attributes still form a super key.

This algorithm, determines only one key out of the possible candidate keys for R; the key returned depends on the order in which attributes are removed from R in step 2.

Database Management Systems

Finding Candidate Keys

We can determine the candidate keys of a given relation using the following steps-

Step-01: Determine all essential attributes of the given relation.

- Essential attributes are those attributes which are not present on RHS of any functional dependency.
- Essential attributes are always a part of every candidate key.
- This is because they can not be determined by other attributes.

Step-02: The remaining attributes of the relation are non-essential attributes. This is because they can be determined by using essential attributes.

Now, following two cases are possible-

Database Management Systems

Finding Candidate Keys

We can determine the candidate keys of a given relation using the following steps-

Case-01: If all essential attributes together can determine all remaining non-essential attributes, then-

- The combination of essential attributes is the candidate key.
- It is the only possible candidate key.

Case-02: If all essential attributes together can not determine all remaining non-essential attributes, then-

- The set of essential attributes and some non-essential attributes will be the candidate key(s).
- In this case, multiple candidate keys are possible.
- To find the candidate keys, we check different combinations of essential and non-essential attributes.

Database Management Systems

Practice Problem: Finding Candidate Keys

Problem-01: Let R = (A, B, C, D, E, F) be a relation scheme with the following dependencies-

C → F
E → A
EC → D
A → B

Which of the following is a key for R?
CD
EC
AE
AC

Also, determine the total number of candidate keys and super keys.

Database Management Systems

Practice Problem: Finding Candidate Keys

Solution- We will find candidate keys of the given relation in the following steps-

Step-01: Determine all essential attributes of the given relation.
Essential attributes of the relation are- C and E. So, attributes C and E will definitely be a part of every candidate key.

Step-02: Now, We will check if the essential attributes together can determine all remaining non-essential attributes. To check, we find the closure of CE.
So, we have-

$\{ CE \}^+$
$= \{ C , E \}$
$= \{ C , E , F \}$ (Using $C \rightarrow F$)
$= \{ A , C , E , F \}$ (Using $E \rightarrow A$)
$= \{ A , C , D , E , F \}$ (Using $EC \rightarrow D$)
$= \{ A , B , C , D , E , F \}$ (Using $A \rightarrow B$)

Practice Problem: Finding Candidate Keys

We conclude that CE can determine all the attributes of the given relation.
So, CE is the only possible candidate key of the relation.

Thus, Option (B) is correct.

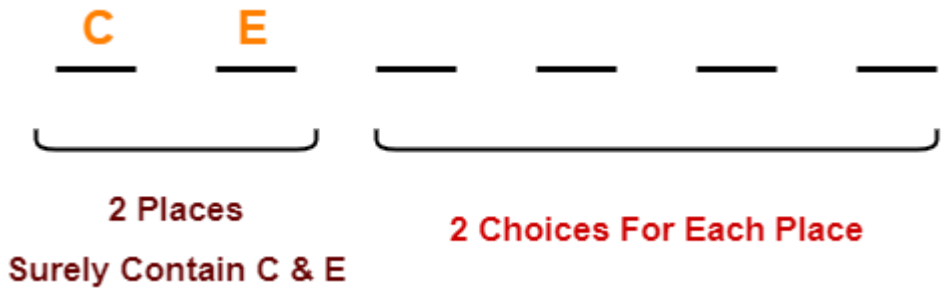
Total Number of Candidate Keys- Only one candidate key CE is possible.

Database Management Systems

Practice Problem: Finding Candidate Keys

Total Number of Super Keys- There are total 6 attributes in the given relation of which-

- There are 2 essential attributes- C and E.
- Remaining 4 attributes are non-essential attributes.
- Essential attributes will be definitely present in every key.
- Non-essential attributes may or may not be taken in every super key.



So, number of super keys possible = 2 x 2 x 2 x 2 = 16.
Thus, total number of super keys possible = 16.

Practice Problem: Finding Candidate Keys

Practice Problem-02:

Let R = (A, B, C, D, E) be a relation scheme with the following dependencies-

$AB \rightarrow C$

$C \rightarrow D$

$B \rightarrow E$

Determine the total number of candidate keys and super keys.

Practice Problem-03: Consider the relation scheme R(E, F, G, H, I, J, K, L, M, N) and the set of functional dependencies-

$\{ E, F \} \rightarrow \{ G \}$

$\{ F \} \rightarrow \{ I , J \}$

$\{ E, H \} \rightarrow \{ K, L \}$

$\{ K \} \rightarrow \{ M \}$

$\{ L \} \rightarrow \{ N \}$

Determine the total number of candidate keys and super keys.

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu

Database Management Systems

Properties of Relational Decomposition

Prof. Ruby Dinakar & Prof. Nivedita

Department of Computer Science and Engineering

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the below resource and persons:
- Author slides from “Fundamentals of Database Systems”, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

15.1 Further topics in Functional Dependencies: Inference Rules, Equivalence and Minimal Cover

15.2 Properties of Relational Decomposition

15.3 Algorithms for Relational Database Schema Design

15.4 About Nulls, Dangling Tuples and Alternative Relational Designs

Designing A Set Of Relations

- **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 Algorithm 15.3).

Designing A Set Of Relations

- **Goals:**
  - Lossless join property (a must)
    - Algorithm 15.3 tests for general losslessness.
  - Dependency preservation property
    - Observe as much as possible
    - Algorithm 15.5 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)

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.
- Decomposition:
  - The process of decomposing the universal relation schema R into a set of relation schemas D = {R1,R2, …, Rm} that will become the relational database schema by using the functional dependencies.
- Attribute preservation condition:
  - Each attribute in R will appear in at least one relation schema R_i in the decomposition so that no attributes are "lost".
- Another goal of decomposition is to have each individual relation R_i in the decomposition D be in BCNF or 3NF.
- Additional properties of decomposition are needed to prevent from generating spurious tuples

Properties of Relational Decompositions (Cont.)

2) Dependency Preservation Property of a Decomposition:

•Definition: Given a set of dependencies F on R, the **projection** of F on R_i, denoted by $\pi_{Ri}(F)$ where R_i is a subset of R, is the set of dependencies $X \rightarrow Y$ in F^+ such that the attributes in X ∪ 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.

- 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 14.13a and Fig 14.12)

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

Properties of Relational Decompositions (Cont.)

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

- Definition: Lossless join property: a decomposition D = {R1, R2, ..., Rm} of R has the **lossless (non-additive) 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: * ($\pi_{R1}(r)$, ..., $\pi_{Rm}(r)$) = 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**"
- Non-additive join term means no spurious tuples results after the application of PROJECT and JOIN operations.

Properties of Relational Decompositions (Cont.)

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

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

1. Create an initial matrix S with one row i for each relation Ri in D, and one column j for each attribute Aj in R.

2. Set S(i,j):=bij for all matrix entries. (* each bij is a distinct symbol associated with indices (i,j) *).

3. For each row i representing relation schema Ri

{for each column j representing attribute Aj

{if (relation Ri includes attribute Aj) then set S(i,j):= aj;};};

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

Properties of Relational Decompositions (Cont.)

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

Algorithm 15.3: Testing for Lossless Join Property (continued)

4. Repeat the following loop until a complete loop execution results in no changes to S
{for each functional dependency X →Y in F
{for all rows in S *which have the same symbols* in the columns corresponding to attributes in X
{make the symbols in each column that correspond to an attribute in Y be the same in all these rows as follows:
If any of the rows has an "a" symbol for the column, set the other rows to that *same* "a" symbol in the column.
If no "a" symbol exists for the attribute in any of the rows, choose one of the "b" symbols that appear in one of the rows for the attribute and set the other rows to that same "b" symbol in the column ;};
};
};

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

Example-1

Figure 15.1 Nonadditive join test for n-ary decompositions.
(a) Case 1: Decomposition of EMP_PROJ into EMP_PROJ1 and EMP_LOCS fails test.
(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}

	Ssn	Ename	Pnumber	Pname	Plocation	Hours
R_1	b_{11}	a_2	b_{13}	b_{14}	a_5	b_{16}
R_2	a_1	b_{22}	a_3	a_4	a_5	a_6

(No changes to matrix after applying functional dependencies)

(b)

EMP

Ssn	Ename

PROJECT

Pnumber	Pname	Plocation

WORKS_ON

Ssn	Pnumber	Hours

Properties of Relational Decompositions (Cont.)

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

(c) R = {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	~~b_{32}~~ a_2	a_3	~~b_{34}~~ a_4	~~b_{35}~~ a_5	a_6

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

Example-2

R(S,A,I,P)
F = {S -> A, SI -> P}

	S	A	I	P
R1	a1	a2	b13	b14
R2	a1	b32	a3	a4

	S	A	I	P
R1	a1	a2	b13	b14
R2	a1	**a2**	a3	a4

Lossless (Row R2 has all a's)

Database Management Systems

Example-3

R(A,B,C,D,E)

F = {A -> C, B -> C, C -> D, DE -> C, CE -> A}

R1(AD), R2(AB), R3(BE), R4(CDE), R5(AE)

	A	B	C	D	E
R1	a1	b12	b13	a4	b15
R2	a1	a2	b23	b24	b25
R3	b31	a2	b33	b34	a5
R4	b41	b42	a3	a4	a5
R5	a1	b52	b53	b54	a5

Example-2

R(A,B,C,D,E)
FD = A -> C

	A	B	C	D	E
R1	a1	b12	**a3**	a4	b15
R2	a1	a2	**a3**	b24	b25
R3	b31	a2	b33	b34	a5
R4	b41	b42	a3	a4	a5
R5	a1	b52	**a3**	b54	a5

Example-2

R(A,B,C,D,E)
FD = B -> C

	A	B	C	D	E
R1	a1	b12	**a3**	a4	b15
R2	a1	a2	**a3**	b24	b25
R3	b31	a2	**a3**	b34	a5
R4	b41	b42	a3	a4	a5
R5	a1	b52	**a3**	b54	a5

Example-2

R(A,B,C,D,E)
FD = C -> D

	A	B	C	D	E
R1	a1	b12	**a3**	a4	b15
R2	a1	a2	**a3**	**a4**	b25
R3	b31	a2	**a3**	**a4**	a5
R4	b41	b42	a3	a4	a5
R5	a1	b52	**a3**	**a4**	a5

Example-2

R(A,B,C,D,E)
FD = CE -> A

	A	B	C	D	E
R1	a1	b12	**a3**	a4	b15
R2	a1	a2	**a3**	**a4**	b25
R3	**a1**	a2	**a3**	**a4**	a5
R4	**a1**	b42	a3	a4	a5
R5	a1	b52	**a3**	**a4**	a5

Properties of Relational Decompositions (Cont.)

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

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

Properties of Relational Decompositions (Cont.)

5)Successive Non-additive Join Decomposition:

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

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu

Database Management Systems

Algorithms for Relational Database Schema Design

Prof. Ruby Dinakar & Prof. Nivedita

Department of Computer Science and Engineering

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the below resource and persons:
- Author slides from “Fundamentals of Database Systems”, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

15.1 Further topics in Functional Dependencies: Inference Rules, Equivalence and Minimal Cover

15.2 Properties of Relational Decomposition

15.3 Algorithms for Relational Database Schema Design

15.4 About Nulls, Dangling Tuples and Alternative Relational Designs

Database Management Systems

Introduction

- In this section we create 2 algorithms :

- First algorithm Decomposes a universal relation into dependency –preserving 3NF relations that also possess the non additive join property.

- Second algorithm Decomposes a universal relation schema into BCNF schemas that possess the non additive join property.

- *It is not possible to design an algorithm to produce BCNF relations that satisfy both dependency preservation and non additive join decomposition.*

Database Management Systems

Introduction

- By now we know that it is not possible to have all 3 of the following:

i) guaranteed nonlossy (non-additive) design

ii) guaranteed dependency preservation

iii) all relations in BCNF

- The first condition is must and can not be compromised.
- The second condition is desirable and but not must and may have to be relaxed if we insist on achieving BCNF.
- The original lost FDs can be recovered by a JOIN operation over the results of decomposition.

Algorithms for Relational Database Schema Design

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

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

1. Find a minimal cover G for F (use Algorithm 15.0).

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. If none of the relation schemas in D contains a key of R, then create one more relation schema in D that contains attributes that form a key of R. *(Use Algorithm 15.4a to find the key of R)*

Example 1 of Algorithm 15.4

Consider the following universal relation: U{SSN, PNO, SAL, DNO, PHONE,PNAME,PLOC)

FD1: SSN -> {SAL,PHONE,DNO}

FD2:PNO->{PNAME,PLOC}

FD3:SSN,PNO->{SAL,PHONE,DNO,PNAME,PLOC}

STEP1: by applying minimal cover algorithm 15.2, we see that FD3 completely redundant.
Hence minimal cover G includes only FD1 and FD2

STEP2: Produce relation R1 and R2 as R1(SSN,SAL,PHONE,DNO) and R2(PNO,PNAME,PLOC)

STEP3: Generate a relation corresponding to the key {SSN,PNO} of U. Hence, the resulting design contains:

R1(SSN,SAL,PHONE,DNO)
R2(PNO,PNAME,PLOC)
R3(SSN, PNO)

The design achieves both the desirable properties of dependency preservation and non-additive join.

Example 2 of Algorithm 15.4

Consider the following universal relation: U{PID, COUNTRY, LOT#, AREA)

FD1: PID -> {LOT#, COUNTRY, AREA}

FD2:LOT#, COUNTRY->{AREA, PID}

FD3:AREA->{COUNTRY}

STEP1: In the set F, PID -> AREA can be inferred from PID -> LOT#,COUNTRY and LOT#,COUNTRY ->AREA; hence PID -> AREA by transitivity and is therefore redundant. Thus one possible minimal cover is PID-> LOT#, COUNTRY , LOT#,COUNTRY->AREA,PID , AREA ->COUNTRY

STEP2: Produce relation **R1(P,L,C) R2(L,C,A,P) R3(A,C)**

STEP3: We find that R3 is subsumed by R2 (i.e. R3 is always a projection of R2 and R1 is a projection of R2 as well) Hence both of those relations are redundant. Thus the 3NF schema that achieves both of the desirable properties is (after removing redundant relations)

R2(L,C,A,P)

Example 2 of Algorithm 15.4 (second approach)

Consider the following universal relation: U{PID, COUNTRY, LOT#, AREA)
FD1: PID -> {LOT#, COUNTRY, AREA}
FD2:LOT#, COUNTRY->{AREA, PID}
FD3:AREA->{COUNTRY}

STEP1: In the set F, LOT#,COUNTRY -> AREA may be redundant because LOT#,COUNTRY -> PID and PID -> AREA by transitivity.
Also, PID -> COUNTRY may be considered to be redundant because PID -> AREA and AREA -> COUNTRY implies PID -> COUNTRY transitivity. This gives a different minimal cover as PID->LOT#,AREA LOT#,COUNTRY -> PID AREA ->COUNTRY

STEP2: Produce relation **R1(P,L,A R2(L,C,P) R3(A,C)**

STEP3: All FDs in the original set F are preserved. We can not eliminate any relations. . Hence, the resulting design contains: R1, R2 and R3

NOTE: it is possible to generate alternate 3NF designs by starting from the same set of FDs.

- **Design of BCNF Schemas**

Algorithm 15.5: Relational Decomposition into BCNF with Lossless (non-additive) join property

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

1. Set D := {R};

2. While there is a relation schema Q in D that is not in BCNF

```
do {
        choose a relation schema Q in D that is not in BCNF;
        find a functional dependency X → Y in Q that violates BCNF;
        replace Q in D by two relation schemas (Q - Y) and (X u Y);
};
```

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

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu

Database Management Systems

About Nulls, Dangling Tuples and Alternative Relational Designs

Prof. Ruby Dinakar & Prof. Nivedita

Department of Computer Science and Engineering

Slides Credits for all PPTs of this course

- The slides/diagrams in this course are an **adaptation**, **combination**, and **enhancement** of material from the below resource and persons:
- Author slides from “Fundamentals of Database Systems”, Ramez Elamsri, Shamkant B Navathe, Pearson, 7th Edition, 2017.

15.1 Further topics in Functional Dependencies: Inference Rules, Equivalence and Minimal Cover

15.2 Properties of Relational Decomposition

15.3 Algorithms for Relational Database Schema Design

15.4 About Nulls, Dangling Tuples and Alternative Relational Designs

Problems with NULL values

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

.

Problems with Null Values and Dangling Tuples (Cont.)

(a)

EMPLOYEE

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

DEPARTMENT

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

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

Database Management Systems

Problems with Null Values and Dangling Tuples (Cont.)

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

Problems with Null Values and Dangling Tuples (Cont.)

Problems with Dangling Tuples

- Consider the decomposition of EMPLOYEE into EMPLOYEE_1 and EMPLOYEE_2 as shown in Figure 15.3 (a) and !5.3 (b).
- Their NATURAL JOIN yields the original relation EMPLOYEE in Figure 15.2(a).
- We may use the alternative representation, shown in Figure 15.3(c), where we *do not include a tuple* in EMPLOYEE_3 if the employee has not been assigned a department (instead of including a tuple with NULL for Dnum as in EMPLOYEE_2).
- If we use EMPLOYEE_3 instead of EMPLOYEE_2 and apply a NATURAL JOIN on EMPLOYEE_1 and EMPLOYEE_3, the tuples for Berger and Benitez will not appear in the result; these are called **dangling tuples** in EMPLOYEE_*1 because they are represented in only one of the two relations that represent employees and hence they are lost if we apply an inner join operation.*

Problems with Null Values and Dangling Tuples (Cont.)

(a) EMPLOYEE_1

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

(b) EMPLOYEE_2

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

(c) EMPLOYEE_3

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

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

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 15.5).

Database Management Systems

Summary of Algorithms for Relational Database Schema Design (1)

PES UNIVERSITY ONLINE

Table 15.1 Summary of the Algorithms Discussed in This Chapter

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

Database Management Systems

Summary of Algorithms for Relational Database Schema Design (2)

PES UNIVERSITY ONLINE

Table 15.1 Summary of the Algorithms Discussed in This Chapter

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

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu

Database Management Systems

Numerical Examples

Prof. Ruby Dinakar & Prof. Nivedita

Dept of Computer Science and Engineering

Numerical Examples on Closure

1) Given relational schema **R(P Q R S T U V)** having following attribute P Q R S T U and V, also there is a set of functional dependency denoted by **FD = { P->Q, QR->ST, PTU->V }.**
Determine Closure of **$(QR)^+$ and $(PR)^+$**

2. Given relational schema **R(P Q R S T)** having following attributes P Q R S and T, also there is a set of functional dependency denoted by **FD = { P->QR, RS->T, Q->S, T-> P }.**
Determine Closure of **$(T)^+$**

Numerical Examples on Normalization:2NF

1. Given a relation R(P, Q, R, S, T) and Functional Dependency set FD = { PQ → R, S → T }, determine whether the given R is in 2NF? If not convert it into 2 NF.

Solution:

Step 1: Let us construct an arrow diagram on R using FD to calculate the candidate key.

Step 2: Find out candidate key from the above diagram using closure approach.

PQS + = PQSRT

Since the closure of PQS contains all the attributes of R, hence **PQS is Candidate Key**

Step 3: find prime and non prime attribute

Since R has 5 attributes: - P, Q, R, S, T and Candidate Key is PQS, Therefore, prime attributes (part of candidate key) are P, Q, and S while a non-prime attribute is R and T

Numerical Examples on Normalization: 2NF

Step 4: Check the existing FDs.

a) FD: PQ → R does not satisfy the definition of 2NF, that non-prime attribute(R) is partially dependent on part of candidate key PQS.

b) FD: S → T does not satisfy the definition of 2NF, as a non-prime attribute(T) is partially dependent on candidate key PQS (i.e., key should not be broken at any cost).

Hence, FD PQ → R and S → T, the above table R(P, Q, R, S, T) is not in 2NF

Step 5: Decompose the table (Convert the table R(P,Q,R,S,T) in 2NF)

Since due to FD: PQ → R and S → T, our table was not in 2NF, let's decompose the table

R1(P, Q, R) (Now in table R1 FD: PQ → R is Full F D, hence R1 is in 2NF)

R2(S, T) (**Now in table R2 FD: S → T is Full F D, hence R2 is in 2NF) And create one table for the key, since the key is PQS. R3(P, Q, S)**

Finally, the decomposed tables which is in 2NF are:

a) R1(P, Q, R)

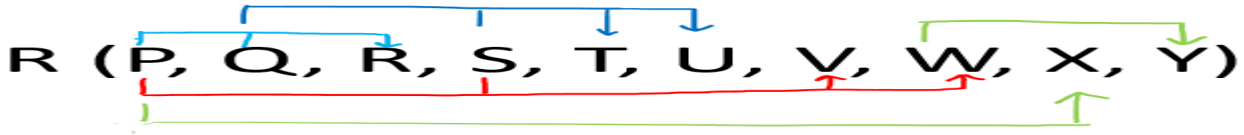
b) R2(S, T)

c) R3(P, Q, S)

2. Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Functional Dependency set FD = { PQ → R, PS → VW, QS → TU, P → X, W → Y }, determine whether the given R is in 2NF? If not convert it into 2 NF.

Solution:

Step 1: Let us construct an arrow diagram on R using FD to calculate the candidate key.



Step 2: Find out candidate key from the above diagram using closure approach.

PQS + = P Q S R T U V W X Y

Since the closure of PQS contains all the attributes of R, hence **PQS is Candidate Key**

Step 3: find prime and non prime attribute

Since R has 10 attributes: - P, Q, R, S, T, U, V, W, X, Y, prime attribute(part of candidate key) are P, Q, and S while non-prime attribute are R, T, U, V, W, X and Y

2. Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Functional Dependency set FD = { PQ → R, PS → VW, QS → TU, P → X, W → Y }, determine whether the given R is in 2NF? If not convert it into 2 NF.

Step 4: Check the existing FDs.

FD: **PQ** → **R** does not satisfy the definition of 2NF, that non-prime attribute(R) is partially dependent on part of candidate key PQS

FD: **PS** → **VW** does not satisfy the definition of 2NF, that non-prime attribute(VW) is partially dependent on part of candidate key PQS

FD: **QS** → **TU** does not satisfy the definition of 2NF, that non-prime attribute(TU) is partially dependent on part of candidate key PQS

FD: **P** → **X** does not satisfy the definition of 2NF, that non-prime attribute(X) are partially dependent on part of candidate key PQS

FD: **W** → **Y** does **not violate** the definition of 2NF, as the non-prime attribute(Y) is dependent on the non-prime attribute(W), which is not related to the definition of 2NF.

Step 5: Decompose the table

2. Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Functional Dependency set FD = { PQ → R, PS → VW, QS → TU, P → X, W → Y }, determine whether the given R is in 2NF? If not convert it into 2 NF.

Step 5: Decompose the table
R1(P, Q, R) (Now in table R1 FD: PQ → R is Full F D, hence R1 is in 2NF)
R2(P, S, V, W) (Now in table R2 FD: PS → VW is Full F D, hence R2 is in 2NF)
R3(Q, S, T, U) (Now in table R3 FD: QS → TU is Full F D, hence R3 is in 2NF)
R4(P, X) (Now in table R4 FD : P → X is Full F D, hence R4 is in 2NF)
R5(W, Y) (Now in table R5 FD: W → Y is Full F D, hence R2 is in 2NF)
And create one table for the key, since the key is PQS. **R6(P, Q, S)**

Finally, the decomposed tables which is in 2NF are:
R1(P, Q, R)
R2(P, S, V, W)
R3(Q, S, T, U)
R4(P, X)
R5(W, Y)
R6(P, Q, S)

Numerical Examples on Normalization:2NF

3. Practice Example

Given a relation R(A, B, C, D, E) and Functional Dependency set FD = { A → B, B → E, C → D}, determine whether the given R is in 2NF? If not convert it into 2 NF.

Numerical Examples on Normalization:3NF

1. Given a relation R(X, Y, Z) and Functional Dependency set FD = { X → Y and Y → Z }, determine whether the given R is in 3NF? If not convert it into 3 NF.

Step 1: Let us construct an arrow diagram on R using FD to calculate the candidate key.

Step 2: find out candidate key:

X + = XYZ

Since the closure of X contains all the attributes of R, hence **X is Candidate Key**

Step 3: (First check for 2NF) find prime and non prime attributes:

X is a prime and Y, Z are non prime

Step 4: Check the existing FDs

FD: X → Y is in 2NF (as Key is not breaking and its Fully functional dependent)

FD: Y → Z is also in 2NF(as it does not violate the definition of 2NF)

Hence it is in 2NF

Numerical Examples on Normalization:3NF

1. Given a relation R(X, Y, Z) and Functional Dependency set FD = { X → Y and Y → Z }, determine whether the given R is in 3NF? If not convert it into 3 NF.

Step 5: Check For 3 NF (A relational schema R is said to be in 3NF, First, it should be in 2NF and, no non-prime attribute should be transitively dependent on the Key of the table.)

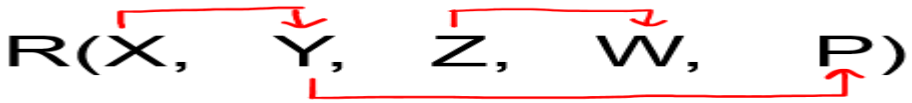
FD: X → Y is in 3NF (as X is a super Key)
FD: Y → Z is not in 3NF (as neither Y is Key nor Z is a prime attribute)
Hence R is not in 3NF.

Step 6: Convert the table R(X, Y, Z) into 3NF:
Decompose the table
FD: Y → Z was creating issue, hence one table R1(Y, Z)
Create one Table for key X, R2(X, Y), since X → Y
Hence decomposed tables which are in 3NF are:
R1(X, Y)
R2(Y, Z)

Numerical Examples on Normalization:3NF

2. Given a relation R(X, Y, Z, W, P) and Functional Dependency set FD = { X → Y, Y → P, and Z → W}, determine whether the given R is in 3NF? If not convert it into 3 NF.

Step 1: Let us construct an arrow diagram on R using FD to calculate the candidate key.



Step 2: find out candidate key:

XZ + = XZYPW

Since the closure of XZ contains all the attributes of R, hence **XZ is Candidate Key**

Step 3: (First check for 2NF) find prime and non prime attributes:

R has 5 attributes: - X, Y, Z, W, P and Candidate Key is XZ, Therefore, prime attribute (part of candidate key) are X and Z while a non-prime attribute are Y, W, and P

Step 4: Check the existing FDs

FD: **X** → **Y** does not satisfy the definition of 2NF, (as key is breaking)

FD: **Z** → **W** does not satisfies the definition of 2NF, (as key is breaking)

2. Given a relation R(X, Y, Z, W, P) and Functional Dependency set FD = { X → Y, Y → P, and Z → W}, determine whether the given R is in 3NF? If not convert it into 3 NF.

Step 5: decompose the relation R for 2NF
R1(X,Y) , R2(Z,W), R3(X, Z), R4(X, Y,P)

Step 6: Now Check For 3 NF (for R4)
FD: X → Y is in NOT in 3NF (as neither X is a super Key nor Y is a prime attribute)
FD: Y → P is not in 3NF (as neither Y is Key nor is a prime attribute)
Hence R4 is not in 3NF.

Step 7: Convert the table R(X, Y, Z,W, P) into 3NF:
Decompose the table (R3)
Hence decomposed tables which are in 3NF are:
R1(X, Y) {Using FD X → Y}
R2(Z, W) {Using FD Z → W}
R3(X, Z) { Using Candidate Key XZ }
R4(Y, P) {Using FD Y → P}

Numerical Examples on Normalization:3NF

3. Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Functional Dependency set FD = { PQ → R, P → ST, Q → U, U → VW, and S → XY}, determine whether the given R is in 3NF? If not convert it into 3 NF.

Step 1: Let us construct an arrow diagram on R using FD to calculate the candidate key.

Step 2: find out candidate key:

PQ + = P Q R S T U X Y V W

Since the closure of XZ contains all the attributes of R, hence **PQ is Candidate Key**

Step 3: (First check for 2NF) find prime and non prime attributes:

R has 10 attributes prime attribute (part of candidate key) are P and Q while a non-prime attribute are R S T U V W X Y V W

Numerical Examples on Normalization: 3NF

3. Given a relation R(P, Q, R, S, T, U, V, W, X, Y) and Functional Dependency set FD = { PQ → R, P → ST, Q → U, U → VW, and S → XY}, determine whether the given R is in 3NF? If not convert it into 3 NF.

Step 4: Check the existing FDs

FD: **PQ** → **R** satisfy the definition of 3NF, as PQ Super Key

FD: **P** → **ST** does not satisfy the definition of 3NF, that neither P is Super Key nor ST is the prime attribute

FD: **Q** → **U** does not satisfy the definition of 3NF, that neither Q is Super Key nor U is a prime attribute

FD: **U** → **VW** does not satisfy the definition of 3NF, that neither U is Super Key nor VW is a prime attribute

FD: **S** → **XY** does not satisfy the definition of 3NF, that neither S is Super Key nor XY is a prime attribute

Database Management Systems

Numerical Examples on Normalization: BCNF

1. Consider a Relation R with five attributes A,B,C,D,E. The functional dependencies are
FD = {A -> BC, C -> DE) Find its in BCNF or not?

Step 1: Candidate key is A.

Step 2: check for each FDs:
A -> BC (determinant is a CK, No violation of BCNF)
C->DE (C is not a CK, hence violation of BCNF)

Step 3: Decomposition of relations
R1(A,BC)
R2(C,D,E)

Hence, R1 and R2 is in BCNF

Database Management Systems

Numerical Examples on Normalization: BCNF

2. Consider a Relation R with four attributes W,X,Y,Z
The functional dependencies are F = {WX -> Y, X -> Z; Y -> W}
Find its in BCNF or not?

Step 1: Candidate key is WX and XY.

Step 2: check for each FDs:
WX -> Y(determinant is a CK, No violation of BCNF)
X->Z (X is not a KEY, hence violation of BCNF)
Y -> W (Y is not a KEY, hence violation of BCNF)

Step 3: Decomposition of relations
R1(W,X,Y)
R2(X,Z)
R3(Y,W)

Hence, R1,R2 and R3 are in BCNF

Database Management Systems

Numerical Examples on Normalization: BCNF

3. Consider the below mentioned relation and Find its in BCNF or not?

Step 1: Candidate key is {author, book title}.

Step 2: These FDs are existed in this relation:
author → nationality
book title → genre, number of pages
And both of the FDs violating BCNF rules

Step 3: Decomposition of relations
R1(author, nationality)
R2(booktitle, genre, number of pages)
R3(Author, booktitle)

Hence, R1,R2 and R3 are in BCNF

Author	Nationality	Book title	Genre	Number of pages
William Shakespeare	English	The Comedy of Errors	Comedy	100
Markus Winand	Austrian	SQL Performance Explained	Textbook	200
Jeffrey Ullman	American	A First Course in Database Systems	Textbook	500
Jennifer Widom	American	A First Course in Database Systems	Textbook	500

Database Management Systems

Numerical Examples on Equivalence of Two Sets of Functional Dependencies

In DBMS, Two different sets of functional dependencies for a given relation may or may not be equivalent. If F and G are the two sets of functional dependencies, then following 3 cases are possible-

Case-01: F covers G ($F \supseteq G$)
Case-02: G covers F ($G \supseteq F$)
Case-03: Both F and G cover each other ($F = G$)

Numerical Examples on Equivalence of Two Sets of Functional Dependencies

Case-01: Determining Whether F Covers G- Following steps are followed to determine whether F covers G or not-

Step-01: Take the functional dependencies of set G into consideration.
For each functional dependency X → Y, find the closure of X using the functional dependencies of set G.

Step-02: Take the functional dependencies of set G into consideration.
For each functional dependency X → Y, find the closure of X using the functional dependencies of set F.

Step-03: Compare the results of Step-01 and Step-02.
If the functional dependencies of set F has determined all those attributes that were determined by the functional dependencies of set G, then it means F covers G.
Thus, we conclude F covers G ($F \supseteq G$) otherwise not.

Case-02: Determining Whether G Covers F-

Following steps are followed to determine whether G covers F or not-

Step-01: Take the functional dependencies of set F into consideration.
For each functional dependency X → Y, find the closure of X using the functional dependencies of set F.

Step-02: Take the functional dependencies of set F into consideration.
For each functional dependency X → Y, find the closure of X using the functional dependencies of set G.

Step-03: Compare the results of Step-01 and Step-02.
If the functional dependencies of set G has determined all those attributes that were determined by the functional dependencies of set F, then it means G covers F.
Thus, we conclude G covers F (G ⊇ F) otherwise not.

Numerical Examples on Equivalence of Two Sets of Functional Dependencies

Case-03: Determining Whether Both F and G Cover Each Other-

If F covers G and G covers F, then both F and G cover each other.
Thus, if both the above cases hold true, we conclude both F and G cover each other (F = G).

Database Management Systems

Numerical Examples on Equivalence of Two Sets of Functional Dependencies

Example 1: A relation R (A , C , D , E , H) is having two functional dependencies sets F and G as shown-

Set F-

$A \rightarrow C$

$AC \rightarrow D$

$E \rightarrow AD$

$E \rightarrow H$

Set G-

$A \rightarrow CD$

$E \rightarrow AH$

Which of the following holds true?

(A) $G \supseteq F$

(B) $F \supseteq G$

(C) $F = G$

(D) All of the above

Numerical Examples on Equivalence of Two Sets of Functional Dependencies

Solution-

CASE 1: Determining whether F covers G-

Step-01: $(A)^+ = \{ A , C , D \}$ // closure of left side of $A \rightarrow CD$ using set G
$(E)^+ = \{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AH$ using set G

Step-02: $(A)^+ = \{ A , C , D \}$ // closure of left side of $A \rightarrow CD$ using set F
$(E)^+ = \{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AH$ using set F

Step-03: Comparing the results of Step-01 and Step-02, we find-
Functional dependencies of set F can determine all the attributes which have been determined by the functional dependencies of set G.
Thus, we conclude F covers G i.e. $F \supseteq G$.

Solution-

CASE 2: Determining whether G covers F-

Step-01: $(A)^+ = \{ A , C , D \}$ // closure of left side of $A \rightarrow C$ using set F
$(AC)^+ = \{ A , C , D \}$ // closure of left side of $AC \rightarrow D$ using set F
$(E)^+ = \{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AD$ and $E \rightarrow H$ using set F

Step-02: $(A)^+ = \{ A , C , D \}$ // closure of left side of $A \rightarrow C$ using set G
$(AC)^+ = \{ A , C , D \}$ // closure of left side of $AC \rightarrow D$ using set G
$(E)^+ = \{ A , C , D , E , H \}$ // closure of left side of $E \rightarrow AD$ and $E \rightarrow H$ using set G

Step-03: Comparing the results of Step-01 and Step-02, we find-
Functional dependencies of set G can determine all the attributes which have been determined by the functional dependencies of set F.
Thus, we conclude G covers F i.e. $G \supseteq F$.

Solution-

CASE 3: Determining whether both F and G cover each other-

From Step-01, we conclude F covers G.
From Step-02, we conclude G covers F.
Thus, we conclude both F and G cover each other i.e. F = G.

Thus, Option (D) is correct.

Database Management Systems

Numerical Examples on Equivalence of Two Sets of Functional Dependencies

PES UNIVERSITY ONLINE

Practice examples:

Q 1. Suppose, a relational schema R (A, B, C) and set of functional dependencies F and G are as follow:

F : { A → B,
B → C,
C → A }

G : { A → BC,
B → A,
C → A }

Check the equivalency of functional dependencies F and G.

Q 2. Suppose, a relational schema R (v w x y z) and set of functional dependencies F and G are as follow:

F : { w → x,
wx → y,
z → wy,
z → v }

G :{ w → xy,
z → wx}

Check the equivalency of functional dependencies F and G.

THANK YOU

Prof. Ruby Dinakar & Prof. Nivedita
Department of Computer Science and Engineering

rubydinakar@pes.edu & niveditak@pes.edu