

CSIE30600/CSIEB0290
Database Systems

Lecture 10:
Relational Database
Design I

Outline

- Informal Design Guidelines for Relational Databases
 - Semantics of the Relation Attributes
 - Redundant Information in Tuples and Update Anomalies
 - Null Values in Tuples
 - Spurious Tuples
- Functional Dependencies (FDs)
 - Definition of FD
 - Inference Rules for FDs
 - Equivalence of Sets of FDs
 - Minimal Sets of FDs

Outline

- 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
- General Normal Form Definitions (For Multiple Keys)
- BCNF (Boyce-Codd Normal Form)

CSIE30600/CSIEB0290 Database Systems

RDB Design I 3

Relational Database Design

- Relational database design requires that we find a “good” collection of relation schemas
- A bad design may lead to
 - Repetition of Information
 - Inability to represent certain information
- **Design Goals:**
 - Avoid redundant data
 - Ensure that relationships among attributes are represented
 - Facilitate the checking of updates for violation of database integrity constraints

CSIE30600/CSIEB0290 Database Systems

RDB Design I 4

Informal Design Guidelines (1)

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

CSIE30600/CSIEB0290 Database Systems

RDB Design I 5

Informal Design Guidelines (2)

- We first discuss informal guidelines for good relational design
- Then we discuss formal concepts of **functional dependencies** and **normal forms**
 - 1NF (First Normal Form)
 - 2NF (Second Normal Form)
 - 3NF (Third Normal Form)
 - BCNF (Boyce-Codd Normal Form)
- Additional types of dependencies, further normal forms, relational design algorithms by synthesis are discussed in next lecture.

CSIE30600/CSIEB0290 Database Systems

RDB Design I 6

Measures of Quality

- Making sure **attribute semantics** are **clear**
- **Reducing redundant** information in tuples
- **Reducing NULL** values in tuples
- **Disallowing** possibility of generating **spurious tuples**

Guideline 1

- Design relation schema so that it is easy to explain its meaning
- Each tuple in a relation should represent one entity or relationship instance.
- Do not combine attributes from multiple entity types and relationship types into a single 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.*

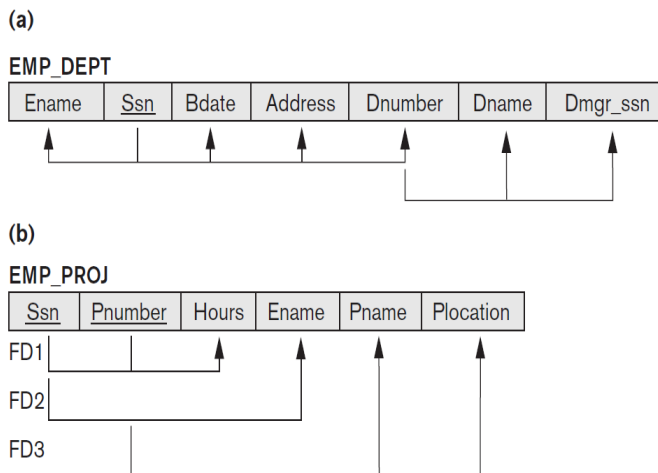
Guideline 1 (cont'd.)

- Example of violating Guideline 1: Figure 14.3

Figure 14.3

Two relation schemas suffering from update anomalies.

- (a) EMP_DEPT and
(b) EMP_PROJ.



Example States for EMP_DEPT and EMP_PROJ

EMP_DEPT							Redundancy
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	

EMP_PROJ						Redundancy	Redundancy
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.

Design Choices: Small vs. Large Schemas

- Which design do you like better? Why?

```
EMPLOYEE(ENAME, SSN, ADDRESS, PNUMBER)
```

```
PROJECT(PNAME, PNUMBER, PMGRSSN)
```

```
EMP_PROJ(ENAME, SSN, ADDRESS, PNUMBER, PNAME, PMGRSSN)
```

- *An employee can be assigned to at most one project, many employees participate in a project*

What's wrong?

```
EMP(ENAME, SSN, ADDRESS, PNUM, PNAME, PMGRSSN)
```

- The description of the project (the name and the manager of the project) is repeated for every employee that works in that department.
- **Redundancy!**
- The project is described redundantly.
- This leads to **update anomalies**.

Redundant Information in Tuples and Update Anomalies

- If information is stored **redundantly**
 - Wastes storage
 - Causes problems with **update anomalies**
- Types of update anomalies:
 - **Insertion** anomalies
 - **Deletion** anomalies
 - **Modification** anomalies

CSIE30600/CSIEB0290 Database Systems

RDB Design I 13

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.

CSIE30600/CSIEB0290 Database Systems

RDB Design I 14

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 an 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.

Guideline 2

- Design base relation schemas so that **no update anomalies** are present in the relations
- If any anomalies are present:
 - Note them clearly
 - Make sure that the programs that update the database will operate correctly

NULL Values in Tuples

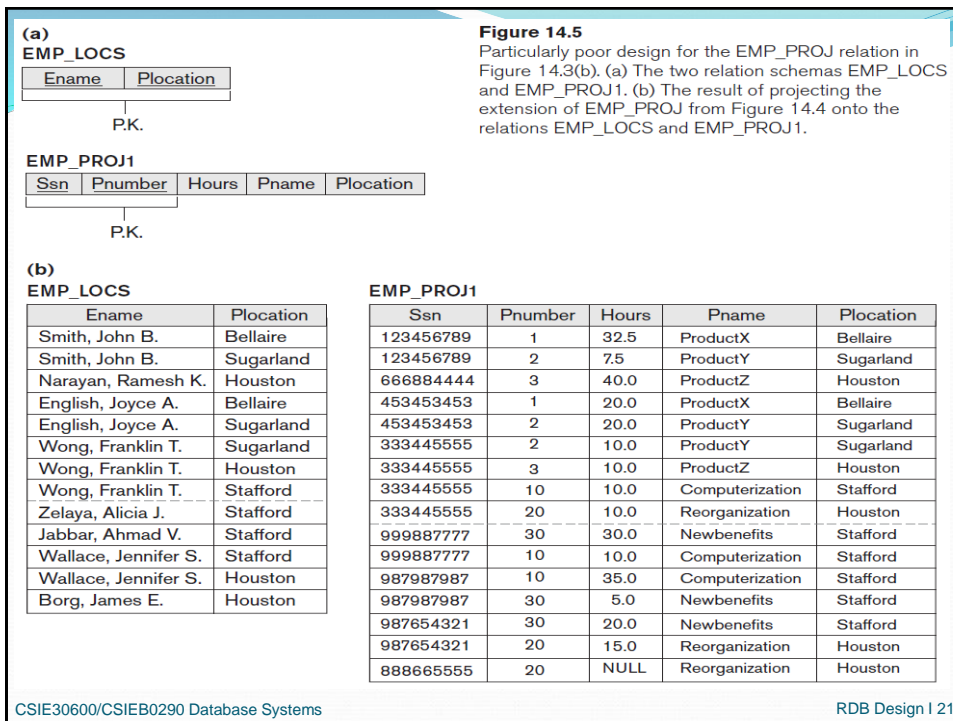
- Some designers may group many attributes together into a “fat” relation
 - Can end up with many NULLs
- Problems with NULLs
 - Wasted storage space
 - Problems understanding meaning

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
- If NULLs are unavoidable:
 - Make sure that they apply in **exceptional cases only**, not to a majority of tuples
- Reasons for NULL s:
 - Attribute **not applicable** or **invalid**
 - Attribute value **unknown** (may exist)
 - Value known to exist, but **unavailable**

Spurious(偽、假) Tuples

- Bad schema designs may result in erroneous results for certain JOIN operations
- Figure 14.5(a)
 - Relation schemas EMP_LOCS and EMP_PROJ₁
- NATURAL JOIN
 - Result produces many more tuples than the original set of tuples in EMP_PROJ
 - Called **spurious tuples**
 - Represent spurious information that is not valid

**Figure 14.5**

Particularly poor design for the EMP_PROJ relation in Figure 14.3(b). (a) The two relation schemas EMP_LOCS and EMP_PROJ1. (b) The result of projecting the extension of EMP_PROJ from Figure 14.4 onto the relations EMP_LOCS and EMP_PROJ1.

Guideline 4

- Design relation schemas to be joined with equality conditions on attributes that are appropriately related
 - Guarantees that **no spurious tuples** are generated
- The "**lossless join**" property is used to guarantee meaningful results for join operations (more about this later)
- **Avoid** relations that contain matching attributes that are **NOT** (foreign key, primary key) combinations

Summary of Design Guidelines

- Anomalies cause redundant work to be done
- Waste of storage space due to NULLs
- Difficulty of performing operations and joins due to NULL values
- Generation of invalid and spurious data during joins

- A good design should avoid all problems above.

Functional Dependencies (FDs)

- Formal tool for analysis of relational schemas
- Enables us to detect and describe some of the above-mentioned problems in precise terms
- Theory of functional dependency

Definition of FDs

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

CSIE30600/CSIEB0290 Database Systems

RDB Design I 25

Definition of FDs (2)

- $X \rightarrow Y$ holds if whenever two tuples have the same value for X, they **must have** the same value for Y
 - For any two tuples t_1 and t_2 in any relation instance $r(R)$: If $t_1[X]=t_2[X]$, then $t_1[Y]=t_2[Y]$
- $X \rightarrow Y$ in R specifies a **constraint** on all relation instances $r(R)$
- Written as $X \rightarrow Y$; 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

CSIE30600/CSIEB0290 Database Systems

RDB Design I 26

Examples of FD Constraints (1)

- Social security number determines employee name
 - $SSN \rightarrow ENAME$
- Project number determines project name and location
 - $PNUMBER \rightarrow \{PNAME, PLOCATION\}$
- Employee SSN and project number determines the hours per week that the employee works on the project
 - $\{SSN, PNUMBER\} \rightarrow HOURS$

Examples of FD Constraints (2)

- Examples of functional dependencies:
 - employee-number \rightarrow employee-name
 - course-number \rightarrow course-title
 - movieTitle, movieYear \rightarrow length, filmType, studioName
- Examples that are NOT functional dependencies
 - employee-name \rightarrow employee-number \times
two distinct employees can have the same name
 - course-number \rightarrow book \times
a course may use multiple books
 - course-number \rightarrow car-color \times
 - ????

What is functional in a FD?

- $A_1, \dots, A_n \rightarrow B$
- A FD is a function that takes a list of values (A_1, \dots, A_n) and produces a **unique** value B or no value at all (this value can be the NULL value)

x	$f(x)$	x	$g(x)$	x	$h(x)$
1	2	1	2	1	10
2	5	2	2	2	20
4	5	3	5	3	30

- We are looking for **functional** relationships (that **must** occur in a relation) among attribute values

More on FD Constraints

- 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 $t_1[K]=t_2[K]$)

Inference Rules for FDs (1)

- Given a set of FDs F , we can **infer** additional FDs that hold whenever the FDs in F hold
- **Armstrong's inference rules:**
 - IR₁. (**Reflexive**) If Y subset-of X , then $X \rightarrow Y$
 - IR₂. (**Augmentation**) If $X \rightarrow Y$, then $XZ \rightarrow YZ$
 - (Notation: XZ stands for $X \cup Z$)
 - IR₃. (**Transitive**) If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$
- IR₁, IR₂, IR₃ form a **sound** and **complete** set of inference rules
 - These are rules hold
 - All other rules that hold can be deduced from these

CSIE30600/CSIEB0290 Database Systems

RDB Design I 31

Inference Rules for FDs (2)

- Some additional inference rules that are useful:
 - **Decomposition:** If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
 - **Union:** If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
 - **Pseudotransitivity:** If $X \rightarrow Y$ and $WY \rightarrow Z$, then $WX \rightarrow Z$
- The last three inference rules, as well as any other inference rules, can be deduced from IR₁, IR₂, and IR₃ (completeness property)

CSIE30600/CSIEB0290 Database Systems

RDB Design I 32

Example: Using Inference Rules

- Prove that if $X \rightarrow Y$ and $Z \rightarrow W$, then $XZ \rightarrow YW$
 1. $X \rightarrow Y$ (given)
 2. $XZ \rightarrow YZ$ (1 and Augmentation)
 3. $Z \rightarrow W$ (given)
 4. $YZ \rightarrow YW$ (3 and Augmentation)
 5. $XZ \rightarrow YW$ (2, 4, and Transitivity)

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
- *If we know how to compute the closure of any set of attributes, we can test if any given FD $A_1, \dots, A_n \rightarrow B$ follows from a set of FDs F*
 - Compute $\{A_1, \dots, A_n\}^+$
 - If $B \in \{A_1, \dots, A_n\}^+$, then $A_1, \dots, A_n \rightarrow B$

Equivalence of Sets of FDs

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

Minimal Sets of FDs (1)

- 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 remove any dependency** from F and have a set of dependencies that is equivalent to F .
 3. We cannot replace any dependency $X \rightarrow A$ in F with a dependency $Y \rightarrow A$, where Y proper-subset-of X (Y subset-of X) and still have a set of dependencies that is equivalent to F .

Minimal Sets of FDs (2)

- Every set of FDs has an equivalent minimal set
- There can be several equivalent minimal sets
- There is no simple algorithm for computing a minimal set of FDs that is equivalent to a set F of FDs
- To synthesize a set of relations, we assume that we start with a set of dependencies that is a minimal set

Normal Forms Based on Primary Keys

- **Normalization** of relations
- Approaches for relational schema design
 - Perform a conceptual schema design using a conceptual model then map conceptual design into a set of relations
 - Design relations based on external knowledge derived from existing implementation of files or forms or reports

Normalization of Relations (1)

- **Normalization:**
 - Takes a schema through a series of tests
 - Certify whether it satisfies a certain normal form
 - **Decompose** unsatisfactory “bad” relations into smaller “good” relations
- **Normal form:**
 - Conditions that must be satisfied for a relation schema to be in a particular “good” form

Normalization of Relations (2)

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

Desirable Properties of Relational Schemas

- **Nonadditive join property (lossless join)**
 - Extremely critical
- **Dependency preservation property**
 - Desirable but sometimes sacrificed for other factors

Practical Use of Normal Forms

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

Problems with Decompositions

- There are three potential problems to consider:
 - Some **queries** become **more expensive**.
 - e.g., In which project does John work? (EMP2 JOIN X)
 - Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!
 - Checking some dependencies may require joining the instances of the decomposed relations.
- *Tradeoff: Must consider these issues vs. redundancy.*

Keys and Attributes (1)

- A **superkey** of a relation schema $R = \{A_1, A_2, \dots, A_n\}$ is a set of attributes S *subset-of* R with the property that no two tuples t_1 and t_2 in any legal relation state r of R will have $t_1[S] = t_2[S]$
- A **key** K is a **superkey** with the *additional property* that removal of any attribute from K will cause K not to be a superkey any more.

Keys and Attributes(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 the basic (flat) relational model
- Most RDBMSs allow only those relations to be defined that are in First Normal Form

Second Normal Form (1)

- Uses the concepts of FDs, **primary key**
- Definitions
 - **Prime attribute**: An attribute that is member of any candidate key K
 - **Full functional dependency**: a FD $Y \rightarrow Z$ where removal of any attribute from Y means the FD does not hold any more
- Examples:
 - $\{\text{SSN, PNUMBER}\} \rightarrow \text{HOURS}$ is a full FD since neither $\text{SSN} \rightarrow \text{HOURS}$ nor $\text{PNUMBER} \rightarrow \text{HOURS}$ hold
 - $\{\text{SSN, PNUMBER}\} \rightarrow \text{ENAME}$ is not a full FD (it is called a **partial dependency**) since $\text{SSN} \rightarrow \text{ENAME}$ also holds

CSIE30600/CSIEB0290 Database Systems

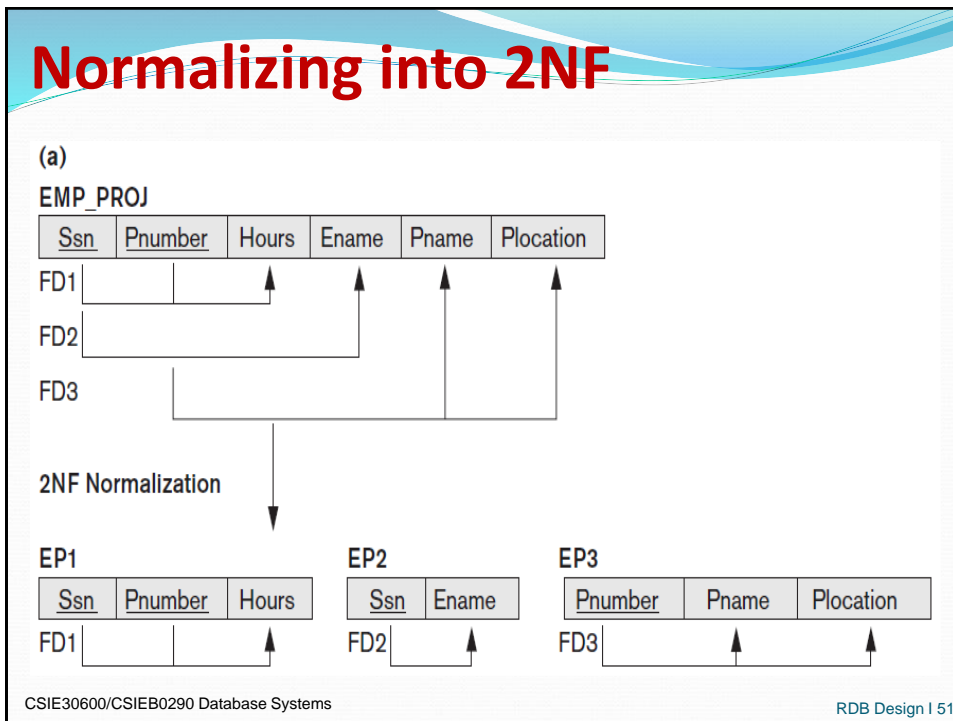
RDB Design I 49

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**

CSIE30600/CSIEB0290 Database Systems

RDB Design I 50



Third Normal Form (1)

- **Transitive dependency** :
 - a FD $X \rightarrow Z$ that can be derived from two FDs $X \rightarrow Y$ and $Y \rightarrow Z$
- **Examples**:
 - $SSN \rightarrow DMGRSSN$ is a **transitive** FD
 - Since $SSN \rightarrow DNUMBER$ and $DNUMBER \rightarrow DMGRSSN$ hold
 - $SSN \rightarrow ENAME$ is **non-transitive**
 - Since there is no set of attributes X where $SSN \rightarrow X$ and $X \rightarrow ENAME$

CSIE30600/CSIEB0290 Database Systems RDB Design I 52

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 $X \rightarrow Y$ and $Y \rightarrow Z$, with X as the primary key, we consider this a problem only if Y is not a candidate key.
 - When Y is a candidate key, there is no problem with the transitive dependency .
 - E.g., Consider EMP (SSN, Emp#, Salary).
 - Here, $SSN \rightarrow Emp\# \rightarrow Salary$ and Emp# is a candidate key.

CSIE30600/CSIEB0290 Database Systems

RDB Design I 53

Normalizing into 3NF

(b)

EMP_DEPT

Ename	<u>Ssn</u>	Bdate	Address	Dnumber	Dname	Dmgr_ssn
-------	------------	-------	---------	---------	-------	----------



3NF Normalization

ED1

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



ED2

<u>Dnumber</u>	Dname	Dmgr_ssn
----------------	-------	----------



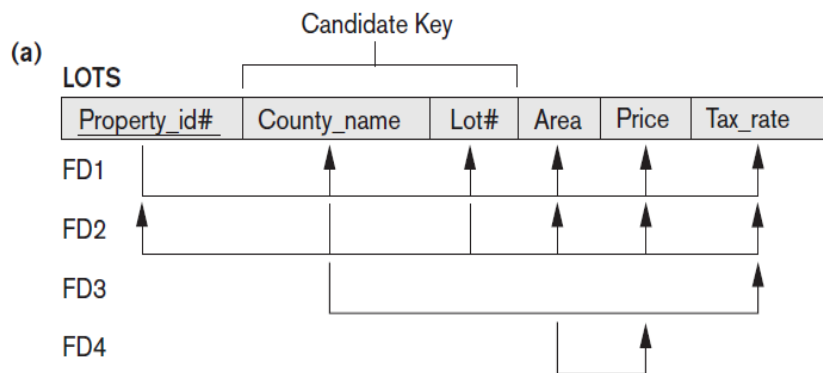
CSIE30600/CSIEB0290 Database Systems

RDB Design I 54

Normalization into 2NF and 3NF

Figure 14.12

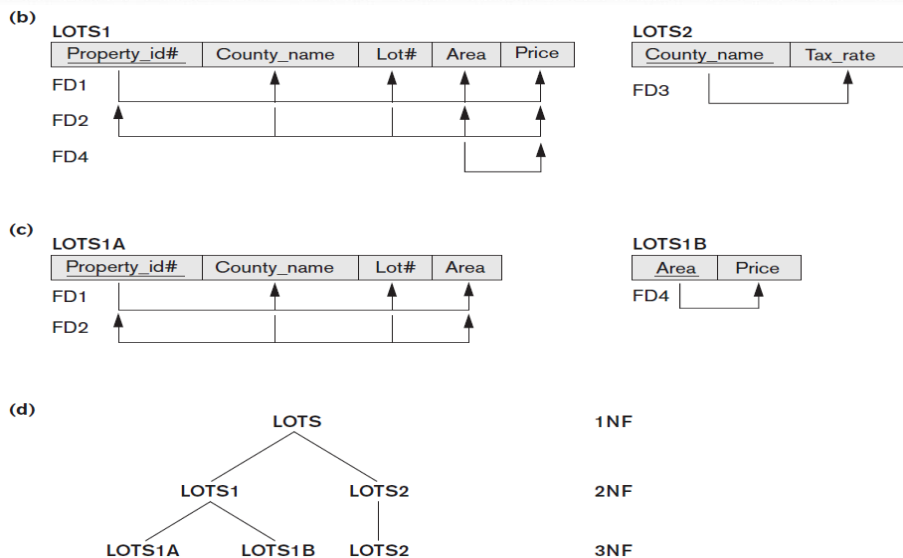
Normalization into 2NF and 3NF. (a) The LOTS relation with its functional dependencies FD1 through FD4. (b) Decomposing into the 2NF relations LOTS1 and LOTS2. (c) Decomposing LOTS1 into the 3NF relations LOTS1A and LOTS1B. (d) Progressive normalization of LOTS into a 3NF design.



CSIE30600/CSIEB0290 Database Systems

RDB Design I 55

Normalization into 2NF and 3NF



CSIE30600/CSIEB0290 Database Systems

RDB Design I 56

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 Definition of Second Normal Form

- The above definitions consider the primary key only
- The following more general definitions take into account relations with multiple candidate keys
- A relation schema R is in **second normal form (2NF)** if every non-prime attribute A in R is not partially dependent on *any key* of R.

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 $X \rightarrow A$ holds in R, then either:
 - (a) X is a superkey of R, or
 - (b) A is a prime attribute of R
- NOTE: Boyce-Codd normal form (slide 61) disallows condition (b) above

CSIE30600/CSIEB0290 Database Systems

RDB Design I 59

Alternative Definition of 3NF

- A relation schema R is in 3NF if **every nonprime attribute** of R meets **both** of the following conditions:
 - It is fully functionally dependent on every key of R.
 - It is nontransitively dependent on every key of R.

CSIE30600/CSIEB0290 Database Systems

RDB Design I 60

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
- **The goal is to have each relation in BCNF (or 3NF)**

CSIE30600/CSIEB0290 Database Systems

RDB Design I 61

Boyce-Codd Normal Form

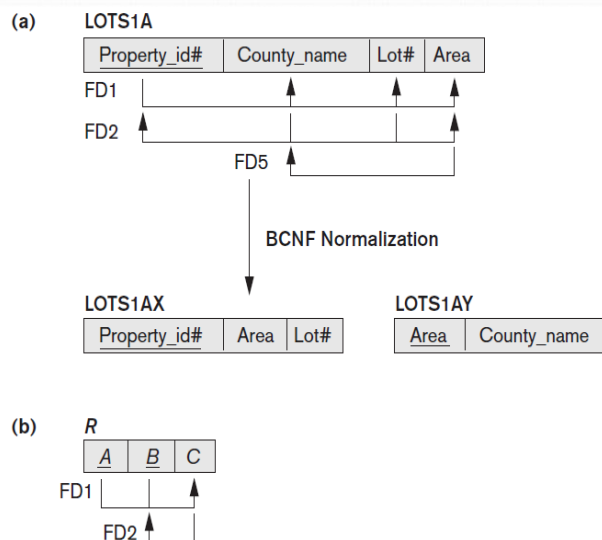


Figure 14.13

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. $C \rightarrow B$.

CSIE30600/CSIEB0290 Database Systems

RDB Design I 62

A relation 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 BCNF by Decomposition(1)

- Two FDs exist in the relation TEACH:
 - $fd_1: \{ \text{student, course} \} \rightarrow \text{instructor}$
 - $fd_2: \text{instructor} \rightarrow \text{course}$
- $\{ \text{student, course} \}$ 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

BCNF by Decomposition (2)

- 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.
- Three possible decompositions for relation TEACH
 - {student, instructor} and {student, course}
 - {course, instructor} and {course, student}
 - {instructor, course} and {instructor, student}

CSIE30600/CSIEB0290 Database Systems

RDB Design I 65

Achieving BCNF by Decomposition (3)

- All three decompositions will lose fd_1 .
 - We have to settle for sacrificing the functional dependency preservation. But we cannot sacrifice the non-additivity property after decomposition.
- Only the 3rd decomposition will not generate spurious tuples after join (and hence has the non-additive property).
- A test to determine whether a binary decomposition (decomposition into two relations) is non-additive (lossless) will be discussed in the next lecture.

CSIE30600/CSIEB0290 Database Systems

RDB Design I 66

Lecture Summary

- Informal Design Guidelines for Relational Databases
- Functional Dependencies (FDs)
 - Definition, Inference Rules, Equivalence of Sets of FDs, Minimal Sets of FDs
- Normal Forms Based on Primary Keys
- General Normal Form Definitions (For Multiple Keys)
- BCNF (Boyce-Codd Normal Form)