# CSc 134 Database Management Systems

## 7. Functional Dependencies and Normalization for Relational Databases

Ying Jin
Computer Science Department
California state University, Sacramento

# Introduction

- What is relational database design?

  The grouping of attributes to form relation schemas

- What are good relational design?
- Formal measures

# Functional Dependencies

- FDs are **constraints** that are derived from
  - meaning and interrelationships of the data attributes
- A functional dependency is a property of the semantics or meaning of the attributes.

# Definition of functional dependency

- A **functional dependency**, denoted by $X \rightarrow Y$, between two sets of attributes X and Y that are subsets of R specifies a constraint on the possible tuples that can form a relation state r of R. The constraint is that, for any two tuples t1 and t2 in r that have t1[X] = t2[X], they must also have t1[Y]=t2[Y].

# FD example

- A set of attributes X *functionally determines* a set of attributes Y if the value of X determines a unique value for Y.
- Social security number functionally determines employee name

  SSN → ENAME

# Notation of Functional Dependencies

- $X \rightarrow Y$
  - function dependency from X to Y
  - Y is functionally dependent on X
  - X: left hand side FD. Y: right hand side FD
- $X \rightarrow Y$ holds if whenever two tuples have the same value for X, they *must have the same value for Y*
- A FD is a property of the relation schema R, not of a particular legal relation state r of R.
- $X \rightarrow Y$ in R specifies a *constraint* on **all** relation instances r(R)

# Examples of FD

- 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

# Infer additional FDs

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

- Given a set of functional dependencies F
  - F= {SSN → ENAME
    PNUMBER → {PNAME, PLOCATION}
    {SSN, PNUMBER} → HOURS }

Infer?

  - {ssn,bdata} → {ename,bdata}
  - Pnumber → pname
  - ssn → hours

# Inference Rules for FDs

- Notation: XZ stands for {X,Z}

Armstrong's inference rules:

**IR1. (Reflexive)**

If $Y \subseteq X$, then $X \rightarrow Y$

**IR2. (Augmentation)**

If $X \rightarrow Y$, then $XZ \rightarrow YZ$

**IR3. (Transitive)**

If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

# Additional Inference Rules

IR 4:(**Decomposition**)

If X → YZ, then X → Y and X → Z

IR 5: (**Union**)

If X → Y and X → Z, then X → YZ

IR6: (**Psuedotransitivity**)

If X → Y and WY → Z, then WX → Z

- Deduced from IR1, IR2, and IR3

# Closure

- **$F^+$: Closure** of F. The set of all dependencies that include F as well as all dependencies that can be inferred from f is called the closure of F.
- **$X^+$: Closure** of X under F. The set of attributes that are functionally determined by X based on F.
- $X^+$ can be calculated by repeatedly applying IR1, IR2, IR3 using the FDs in F

# Algorithm to calculate $X^+$

Determining $X^+$, the closure of x under F

$X^+ := X$;

Repeat

    $oldX^+ := X^+$;

    for each functional dependency Y->Z in F do

    if $Y \subseteq X^+$ then $X^+ := X^+ \cup Z$;

Until ($X^+ = oldX^+$ );

# Example of calculate $X^+$

- F= {SSN-> ENAME,
  PNUMBER -> {PNAME, PLOCATION},
  {SSN,PNUMBER} ->HOURS}

- $\{SSN\}^+ = \{SSN, ENAME\}$
- $\{SSN, PNUMBER\}^+ = \{SSN, PNUMBER, ENAME, PNAME, PLOCATION, HOURS\}$
- $\{PNUMBER\}^+ = \_\_\_?$

# 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

- F and G are **equivalent** if $F^+ = G^+$

Definition: F **covers** G if every FD in G can be inferred from F (i.e., if $G^+ \subseteq F^+$)

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

# Minimal Sets of FDs

A set of FDs is **minimal** if it satisfies the following conditions:

(1) Every dependency in F has a single attribute for its right hand side.

(2) We cannot replace any dependency X -> A in F with a dependency Y -> A, where Y ⊆ X, and still have a set of dependencies that is equivalent to F.

(3) We cannot remove any dependency from F and have a set of dependencies that is equivalent to F.

# Minimal Sets of FDs

- Every set of FDs has an equivalent minimal set
- There can be several equivalent minimal sets
- We can always find at least one minimal set using Algorithm 10.2

# Algorithm 10.2 Finding a Minimal Cover F for a set of functional Dependencies E

1. Set F: = E;
2. Replace each functional dependency X → {A1, A2,…, An} in F by the n functional dependencies X → A1, X → A2,…, X → An
3. For each functional dependency X →A in F
   for each attribute B that is an element of X
   if {{F-{X →A}} U {{x-{B}} →A}} is equivalent to F
   then replace X → A with (X-{B}) → A in F
4. For each remaining functional dependency X→A in F
   if (F-{X →A}) is equivalent to F,
   then remove X →A from F.

# What is normalization?

- **Normalization**: The process of decomposing unsatisfactory relations by breaking up their attributes into smaller relations
  - Use
    - keys
    - FDs

  to certify whether a relation schema is in a particular normal form

# Practical Use of Normal Forms

- **Normalization** is carried out 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.
- **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

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

$(k)^+ =$ ____ ?

- 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.

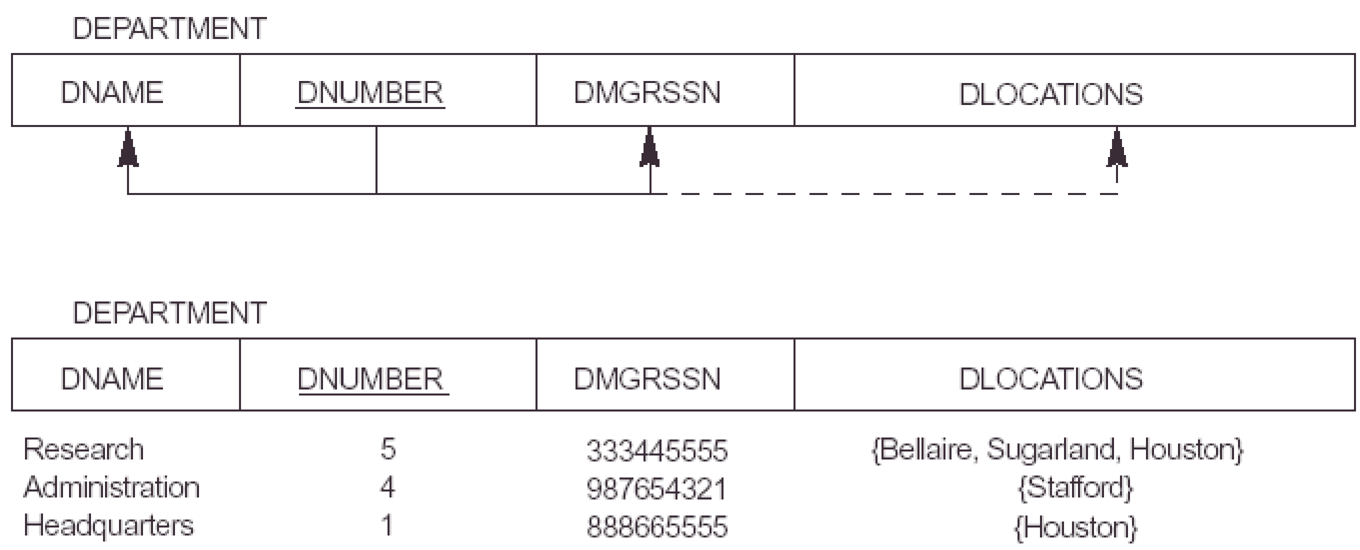
# Definitions of Keys and Attributes Participating in Keys (Cont.)

- 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*.

# First Normal Form

- Disallows composite attributes, multivalued attributes
- Disallows attributes whose values *for an individual tuple* are non-atomic
- Considered to be part of the definition of relation

# Figure 10.8 Normalization into 1NF



# Normalization into 1 NF

- Solution 1 (best)
  - Department(dname,dnumber, dmgrssn)
  - dept_loc(dnumber, dlocation)
- Solution 2
  - department(dnumber,dlocation,dname,dmgrssn)
- Solution 3
  - department(dnumber,dname,dmgrssn, dlocation1,dlocation2,dlocation3)

# Full functional dependency

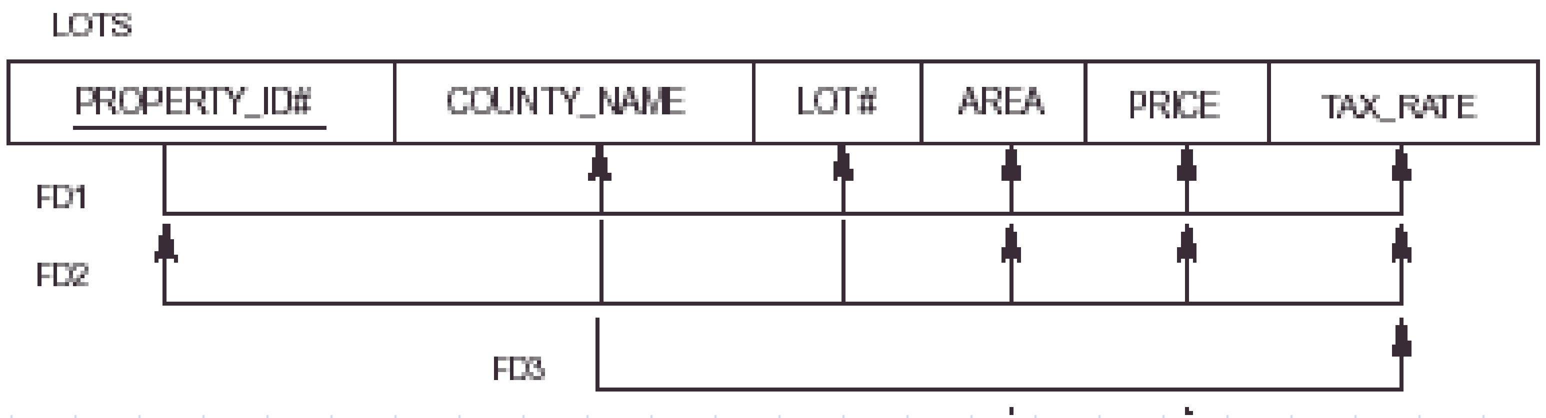
- Full functional dependency
  - a FD Y → Z, where removal of any attribute from Y means the FD does not hold any more
  - e.g. {SSN, PNUMBER} → HOURS
- Partial dependency
  - e.g. {SSN, PNUMBER} → ENAME

# 2 NF

- General definition
- Take into account relations with *multiple candidate keys*
- **Prime attribute**: An attribute that is part of any candidate key
- A relation schema R is in **second normal form** (**2NF**) if <u>every</u> non-prime attribute A in R is fully functionally dependent on <u>every</u> key of R.

# 2NF - Example

- Keys:
  - property_id#
  - {county_name,lot#}
- Violate/satisfy? 2NF



# 3NF

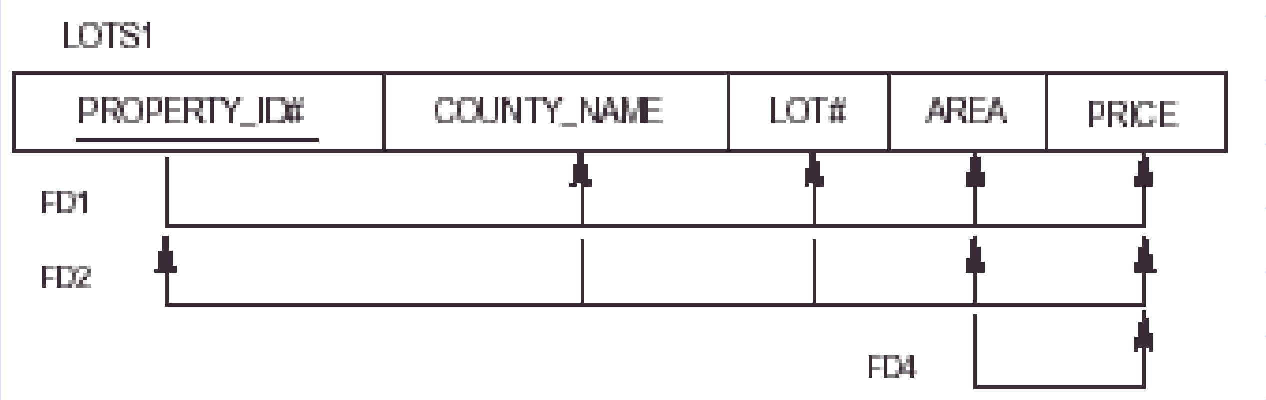
- $X \rightarrow Y$ is **trivial** if $Y \subset X$, otherwise, it is nontrival.
- A relation schema R is in **third normal form** (**3NF**) if, whenever a non-trivial FD $X \rightarrow A$ holds in R, then either:

  (1) X is a superkey of R, or

  (2) A is a prime attribute of R

# 3NF - example

- Keys:
  - property_id#
  - {county_name,lot#}
- Violate/satisfy? 3NF

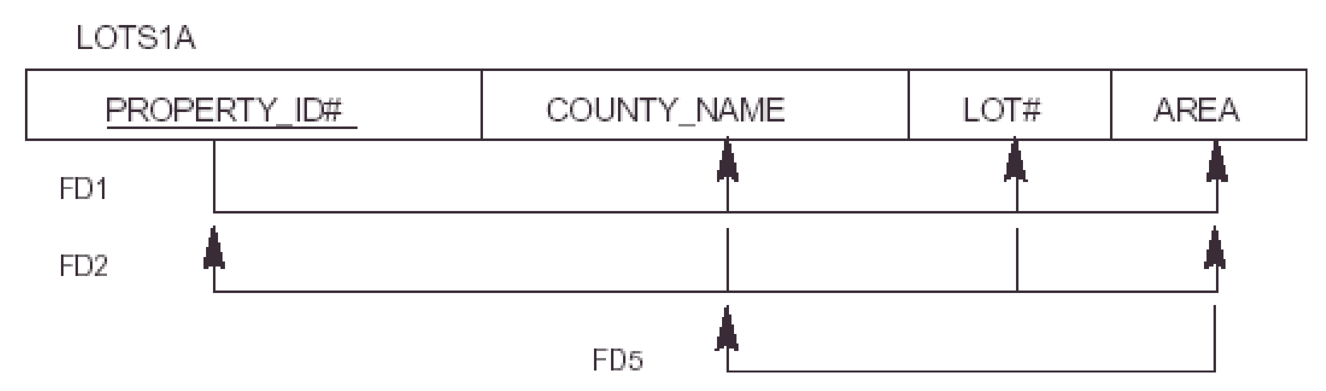


# BCNF (Boyce-Codd Normal Form)

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

# BCNF - example

- Keys:
  - property_id#
  - {county_name,lot#}
- Violate/satisfy? BCNF



These slides are based on the textbook of:

R. Elmaseri and S. Navathe, *Fundamentals of Database Systems*, 7th Edition, Addison-Wesley.