


## Schema Design: Normal Forms, Functional Dependencies

CS 178

1



## Relational Model: Definitions Review

- ❖ Relations/tables, Attributes/Columns, Tuples/rows
  - Attribute domains
- ❖ Superkey
- ❖ Key
  - No two tuples can have the same value in the key attribute
  - Primary key, candidate keys
  - No primary key value can be null
- ❖ Referential integrity constraints
  - Foreign key
- ❖ Next...how to design the schema?

CS 178

2




## Relational Schema Design

- ❖ Logical Level
  - Whether schema has intuitive appeal for users
- ❖ Manipulation level
  - Whether it makes sense from an *efficiency* or *correctness* point of view

CS 178

3



## Functional Dependencies and Normal Forms

- ❖ Guidelines for database schema design: how to design a “good” schema ?
- ❖ Example of a COMPANY database: two possible designs to represent Employees and Department information

S1: EMPLOYEE(LNAME,FNAME,SSN,DNO)  
DEPT(DNUM, DNAME, MGRSSN)

S2:  
EMPDEPT(LNAME,FNAME,SSN,DNUM,DNAME,  
MGRSSN)

Which one is better ?? S1 or S2 ?

CS 178

4

## Functional Dependencies and Normal Forms

- ❖ Informal methods
  - Rules of thumb, intuitive reasoning, experience
- ❖ Formal methods
  - Provable properties
  - Involve concept of **Functional Dependencies**
  - Develop theoretical model to define what we mean by “good schema”

## Informal Guidelines: 1

- ❖ 1: Try to make user interpretation easy
  - S1: EMP(FNAME, LNAME, SSN)  
WORKS\_ON (SSN, PNO)  
PROJECT\_LOC(PNO, PLOC)
  - S2: EMP(FNAME,LNAME,SSN, PNO,PLOC)
- ❖ Perhaps S2 has too much information to absorb per tuple ?

## Informal Guidelines: 2

- ❖ Try to reduce redundancy
  - In S2 in previous example, suppose only few projects
    - PLOC is unnecessarily repeated too often
  - On the other hand, S1 repeats SSN in WORKS\_ON
    - But SSN is a smaller attribute than PLOC (which may be a large string)

## Informal Guidelines: 3

- ❖ Try to avoid update anomalies
  - Avoid having to search through entire table during update operation
    - Insert, delete, update/modify
  - Avoid losing information

## Example

S1: EMPLOYEE (ENAME, SSN, BDATE, ADDR, DNO)  
DEPT (DNUM, MGRSSN, DNAME)  
WORKS\_ON (SSN, PNO, HOURS)  
PROJECT (PNUM, PLOC)

S2: EMP\_DEPT(ENAME,SSN,BDATE, ADDR,DNO, DNAME,MGRSSN)  
EMP\_PROJ (SSN, PNUM, HOURS, ENAME, PNAME, PLOC)

Both schemas have same attributes....  
Problems with S2 ??

## Insertion Anomalies in S2

- ❖ Consider inserting information "John Smith works in Department 5"
  - i.e., tuple should be inserted into EMP\_DEPT  
< John Smith, 23456789,...,5,Research,111223333>
  - What do we need to check in the table to maintain correctness?
    - How to check ?

## Insertion Anomalies in S2

- ❖ What do we need to check in the table to maintain correctness ?
- ❖ Check data is correct – i.e.,
  - Check that all tuples with DNO=5 have DNAME=Research, MGRSSN=111223333
  - Scan the whole relation since DNO is not a key!

## Insertion problems...contd.

- ❖ Consider creating a new department in the Company: DNO=9, DNAME = 'Sales'
- ❖ Only one way to do this: create NULL values for employee info
  - But that means NULL value in primary key (SSN)!!

## Deletion and Modification Anomalies

- ❖ What if we delete the last employee in 'Research' department
  - Eg. (John Smith, 123456789,...,5,'Research',...)
- ❖ Similar cases for Modification
  - Change Manager SSN of department 5 = change for all Department 5 employees ...scan of entire table

CS 178

13

## Informal Guidelines: 4

- ❖ Avoid too many NULL values
  - Space is wasted
  - Problems occur when using aggregate functions like count or sum
  - NULLs can have different intentions
    - Attribute does not apply
    - Value unknown and will remain unknown
    - Value unknown at present

CS 178

14

## Informal Guidelines 5: Spurious Tuples

- ❖ Split a table into smaller tables (with fewer columns in each) – sometimes a better design in our examples
  - How to split ?
- ❖ When reconstructing the “original” data, should not introduce spurious tuples
  - Also called non-additive joins

CS 178

15

## Example: Spurious Tuples

S1: CAR (ID, Make, Color)

S2: CAR1 (ID, Color)


CAR2 (Color, Make)

What happens when we join CAR1 and CAR2 ?

123	Toyota	Blue
456	Audi	Blue
789	Toyota	Red

CS 178


16



S2-Car 1                      S2-Car 2


123	Blue	Blue	Toyota
456	Blue	Blue	Audi
789	Red	Red	Toyota


CS 178 17



123	Blue	Toyota
123	Blue	Audi
456	Blue	Toyota
456	Blue	Audi
789	Red	Toyota

CS 178 18

- 
- ### Summary of Problems
- ❖ Insertion, Deletion, modification anomalies
  - ❖ Too many NULLs
  - ❖ Spurious tuples - called non-additive join
  - ❖ We need a theory of schema design
    - Functional dependencies and normalization
  - ❖ Using functional dependencies define "normal forms" of schema
    - A schema in a "Third Normal Form" will avoid certain anomalies
- CS 178 19



### Functional Dependencies

Describe "Key-Like" Relationships

A key is a set of attributes where:  
If keys match, then the tuples match


A *functional dependency* (FD) is a generalization:  
If an attribute set *determines* another, written  $A \rightarrow B$   
then if two tuples agree on attribute set **A**, they must agree on **B**:

$sid \rightarrow name$

➤ FDs are **independent** of our schema design choice

CS 178 20

## Formal Definition of FD's



Given a relation schema  $R$  and subsets  $X, Y$  of  $R$ :

An instance  $r$  of  $R$  satisfies FD  $X \rightarrow Y$  if,  
for any two tuples  $t1, t2 \in r$ ,

$t1[X] = t2[X]$  implies  $t1[Y] = t2[Y]$

*if they have the same values in  $X$  attributes/columns, they have the same values in  $Y$  attributes/columns*


❖ For an FD to hold for schema  $R$ , it must hold for every possible instance of  $r$

- (Can a DBMS verify this? Can we determine this by looking at an instance?)

CS 178

21

## General Thoughts on Good Schemas



We want all attributes in every tuple to be determined only by the tuple's key attributes, i.e. part of a *superkey* (for key  $X \rightarrow Y$ , a superkey is a "non-minimal"  $X$ )

*What does this say about redundancy?*


But:

- *What about tuples that don't have keys (other than the entire value)?*

CS 178

22

## Sets of Functional Dependencies



❖ Relation EMP-DEPT(SSN, NAME, ADDRESS, DNUMBER, DNAME, MGRSSN)

- Employee info; the dept they are assigned to; their manager's ssn
- Key is SSN


❖ Some obvious functional dependencies

- $\{SSN\} \rightarrow \{NAME, ADDRESS, DNUMBER\}$
- $\{DNUMBER\} \rightarrow \{DNAME, MGRSSN\}$

CS 178

23

## Sets of Functional Dependencies



❖ Some obvious functional dependencies

- $\{SSN\} \rightarrow \{NAME, ADDRESS, DNUMBER\}$
- $\{DNUMBER\} \rightarrow \{DNAME, MGRSSN\}$

❖ From above dependencies, we can infer

- $\{SSN\} \rightarrow \{DNAME, MGRSSN\}$

❖ **Concept of a set of dependencies that can be inferred from the given set**

- Inference rules ?
- Closure:  $F^+$  is all dependencies that can be inferred from  $F$

CS 178

24

## Some Key Questions to ask:

- ❖ Given a set of functional dependencies (properties on the data), what other properties can we infer?
- ❖ What is the formal definition of a key?
- ❖ How can we use the formal framework of Functional dependencies to define a "good schema design"?
- ❖ Can we automate the process (develop algorithms)?

## Armstrong's Axioms: Inferring FDs

Some FDs exist due to others; can compute using **Armstrong's axioms**:

- **Reflexivity:** If  $Y \subseteq X$  then  $X \rightarrow Y$  (*trivial dependencies*)  
name, sid  $\rightarrow$  name
- **Augmentation:** If  $X \rightarrow Y$  then  $XW \rightarrow YW$   
cid  $\rightarrow$  subj so cid, exp-grade  $\rightarrow$  subj, exp-grade
- **Transitivity:** If  $X \rightarrow Y$  and  $Y \rightarrow Z$  then  $X \rightarrow Z$   
cid  $\rightarrow$  crnum and crnum  $\rightarrow$  subj  
so cid  $\rightarrow$  subj

## Armstrong's Axioms Lead to...

- ❖ **Union:** If  $X \rightarrow Y$  and  $X \rightarrow Z$   
then  $X \rightarrow YZ$
- ❖ **Pseudotransitivity:** If  $X \rightarrow Y$  and  $WY \rightarrow Z$   
then  $XW \rightarrow Z$
- ❖ **Decomposition:** If  $X \rightarrow Y$  and  $Z \subseteq Y$   
then  $X \rightarrow Z$

Can prove these from Armstrong's Axioms...

## Next question:

- ❖ Given a set F of functional dependencies, what are all the properties we can infer?
- ❖ Do two sets of functional dependencies, F and G, imply the same set of properties?
- ❖ How to formally define this property?

## Closure of a Set of FD's

**Defn.** Let  $F$  be a set of FD's.

Its *closure*,  $F^+$ , is the set of all FD's:

$\{X \rightarrow Y \mid X \rightarrow Y \text{ is derivable from } F \text{ by Armstrong's Axioms}\}$

## Why Armstrong's Axioms?

Why are Armstrong's axioms (or an equivalent rule set) appropriate for FD's? They are:

- **Consistent:** any relation satisfying FD's in  $F$  will satisfy those in  $F^+$
- **Complete:** if an FD  $X \rightarrow Y$  cannot be derived by Armstrong's axioms from  $F$ , then there exists some relational instance satisfying  $F$  but not  $X \rightarrow Y$

➤ In other words, Armstrong's axioms derive *all* the FD's that should hold

## Equivalence of FD sets

**Defn.** Two sets of FD's,  $F$  and  $G$ , are *equivalent* if their closures are equivalent,  $F^+ = G^+$

e.g., these two sets are equivalent:

$\{XY \rightarrow Z, X \rightarrow Y\}$  and  
 $\{X \rightarrow Z, X \rightarrow Y\}$

- ❖  $F^+$  could contain a huge number of FD's (exponential in the size of the schema?)
- ❖ Would like to have smallest "representative" FD set – the "cover" set for  $F$ 
  - Why?

## Minimal Cover

**Defn.** A FD set  $F$  is *minimal* if:

1. Every FD in  $F$  is of the form  $X \rightarrow A$ , where  $A$  is a single attribute
2. For no  $X \rightarrow A$  in  $F$  is:  
 $F - \{X \rightarrow A\}$  equivalent to  $F$
3. For no  $X \rightarrow A$  in  $F$  and  $Z \subset X$  is:  
 $F - \{X \rightarrow A\} \cup \{Z \rightarrow A\}$  equivalent to  $F$

*we express each FD in simplest form*

*in a sense, each FD is "essential" to the cover*

**Defn.**  $F$  is a *minimum cover* for  $G$  if  $F$  is minimal and is equivalent to  $G$ .

e.g.,

$\{X \rightarrow Z, X \rightarrow Y\}$  is a minimal cover for  
 $\{XY \rightarrow Z, X \rightarrow Y\}$

## Attribute Closures: Is Something Dependent on X?

Defn. The closure of an attribute set X,  $X^+$ , is:

$$X^+ = \bigcup \{Y \mid X \rightarrow Y \in F^+\}$$

- ❖ This answers the question "is Y determined (transitively) by X?";
  - Given values for X, the values of Y are fixed
- ❖ Does  $sid, cid \rightarrow subj, name, exp-grade$ ?

## Computing Attribute Set Closure

- ❖ For attribute set X, compute closure  $X^+$  by:

Closure  $X^+ := X$ ;  
repeat until no change in  $X^+$  {  
  if there is an FD  $U \rightarrow V$  in  $F$   
  such that  $U$  is in  $X^+$   
  then add  $V$  to  $X^+$ }

## Attribute Set Closure and Keys

- ❖ If X is a key over relation scheme R, then what is  $X^+$ ?
  - Formal definition of a Key
- ❖ How to determine the keys for relation R?
  - R is a set of attributes  $\{A_1, A_2, \dots, A_n\}$
  - For each subset S of R, compute  $S^+$ 
    - If  $S^+ = R$  then S is Key
  - What is the "catch" here?
    - NP complete problem
  - Can you improve this?

## Example

- ❖  $R = (C, T, H, R, S)$ 
    - Course (C), Time (T), Hour (H), Room (R), Section (S), Grade (G)
- $C \rightarrow T$                        $CS \rightarrow G$   
 $HS \rightarrow R$                      $HR \rightarrow C$   
 $HT \rightarrow R$

What is the smallest attribute set that must be part of the key?

## Attribute Set Closures: Pruning the search



- ❖ If attribute A does not appear on RHS of any FD, then any key must contain A
- ❖ If X is a key, then anything containing X is a superkey
- ❖ If X is a key, and  $Y \rightarrow X$  is a FD then Y is a key

## Functional Dependencies and Schema Design: Normal Forms



- ❖ Normal forms are properties of relations
- ❖ We say a relation is in xNF if its attributes satisfy certain properties
  - Properties formally defined using functional dependencies
  - For example, test the relation to see if it is in 3NF
  - If not in 3NF, then change design...how?
    - Decomposition

## How to go about designing a good schema ?



- ❖ How to create a 3NF database schema ? (i.e., a good design) ?
- ❖ Ad-hoc approach
  - Create relations intuitively and hope for the best!
- ❖ Formal method – procedure (automated tool)
  - Start with single relation with all attributes
  - Systematically decompose relations that are not in the desired normal form
  - Repeat until all tables are in desired normal form
- ❖ Can decomposition create problems if we are not careful ?
  - (i) Spurious tuples and (ii) lost dependencies

## Decomposition



- ❖ Consider original “bad” schema

Stuff(sid, name, cid, subj, crnum, exp-grade)

- ❖ We could decompose it into

Student(sid, name)  
Course(cid, crnum)  
Subject(crnum, subj)

- ❖ But this decomposition loses information about the relationship between students and courses. Why?

## Lossless Join Decomposition

$R_1, \dots, R_k$  is a *lossless join decomposition* of  $R$  w.r.t. an FD set  $F$  if for every instance  $r$  of  $R$  that satisfies  $F$ ,

$$\Pi_{R_1}(r) \bowtie \dots \bowtie \Pi_{R_k}(r) = r$$

Consider:

sid	name	cid	subj	crnum	exp-grade
1	Sam	570103	SW	cs143	B
23	Dan	550103	DB	cs178	A

What if we decompose on

(sid, name) and (cid, subj, crnum, exp-grade)?

CS 178

41

## Testing for Lossless Join

$R_1, R_2$  is a lossless join decomposition of  $R$  with respect to  $F$  iff *at least one* of the following dependencies is in  $F^+$

$$(R_1 \cap R_2) \rightarrow R_1 - R_2$$

$$(R_1 \cap R_2) \rightarrow R_2 - R_1$$

❖ Set of attributes common to the two tables are key to one of the two table.

CS 178

42

## Example of Lossless and Dependency-Preserving Decompositions

Given relation scheme

$R(\text{name, street, city, st, zip, item, price})$

And FD set

- $\text{name} \rightarrow \text{street, city}$
- $\text{street, city} \rightarrow \text{st}$
- $\text{street, city} \rightarrow \text{zip}$
- $\text{name, item} \rightarrow \text{price}$

Consider the decomposition

$R_1(\text{name, street, city, st, zip})$  and  $R_2(\text{name, item, price})$

➤ Is it lossless?

➤ Is it dependency preserving?

What if we added FD  $\text{street, city} \rightarrow \text{item}$ ?

CS 178

43

## FD's and Keys

❖ Ideally, we want a design s.t. for each nontrivial dependency  $X \rightarrow Y$ ,  $X$  is a superkey for some relation schema in  $R$  and all dependencies are preserved

▪ We just saw that this isn't always possible

❖ What if a dependency is lost during decomposition, but we want to enforce the condition ??

▪ Is there anything in SQL that can help us enforce this dependency condition ?

CS 178

44

## Two Important Normal Forms

**Boyce-Codd Normal Form (BCNF).** For every relation scheme R and for every  $X \rightarrow A$  that holds over R,

- either  $A \in X$  (it is trivial), or
- or  $X$  is a superkey for R

**Third Normal Form (3NF).** For every relation scheme R and for every  $X \rightarrow A$  that holds over R,

- either  $A \in X$  (it is trivial), or
- $X$  is a superkey for R, or
- $A$  is a member of some key for R

## Definitions

### ❖ Relation schema R

- Superkey
- Key
- Candidate key - same as key
- Primary key - a key designated for common use
- Prime attribute - an attribute that belongs to some candidate key
- Non-prime attribute - does not belong to any key

## Partial Dependency

- ❖ A FD  $X \rightarrow Y$  is a partial dependency if there exists an attribute  $A \in X$  such that  $X - A \rightarrow Y$ 
  - $Y$  is partially dependent on  $X$
- ❖ Second Normal Form: Relation is in 2NF if no non-prime attribute is partially dependent on the primary key.

## Problems with Partial Dependency

EMP\_PROJ( SSN, PNUMBER, HOURS, ENAME, PNAME, PLOCATION)

Some FDs:

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

Since  $SSN \rightarrow ENAME$ , ENAME is partially dependent on the primary key {SSN, PNUMBER}

- So why is this a problem ?



- ❖ Insert tuple
  - <987654321, 3, 12, Jones, Sprite, Atlanta>
- ❖ We have insertion anomaly
  - Check if 987654321 is Jones, project 3 is Sprite...
- ❖ We have deletion problem
  - If last tuple with Project #1 is deleted
- ❖ Similarly, we have modification anomaly
  - Smith changes name to Brown



## Transitive Dependencies & 3NF

- ❖ FD  $X \rightarrow Y$  is a transitive dependency in relation R if there exists set of attributes  $Z \in R$  such that
  - $X \rightarrow Z$  and  $Z \rightarrow Y$
  - Z is not a subset of any key of R
- ❖ A relation R is in Third Normal Form if (1) it is in 2NF and (2) no non-prime attribute is transitively dependent on any key.



## Problem with Transitive Dependencies

- ❖ EMP\_DEPT(ENAME, SSN, BDATE, ADDRESS, DNO, DNAME, MGRSSN)
- FDs in relation:
- {SSN}  $\rightarrow$  {DNO}
  - {DNO}  $\rightarrow$  {MGRSSN}
  - {DNO}  $\rightarrow$  {DNAME}
- ❖ Insertion, Deletion, Modification anomalies in above schema



## Problem with 3NF?

- ❖ ADDR\_INFO( CITY, ADDRESS, ZIP)
- {CITY, ADDRESS}  $\rightarrow$  ZIP
- {ZIP}  $\rightarrow$  {CITY}
- Possible keys: {CITY, ADDRESS} or {ADDRESS, ZIP}
- Is it in 3NF?

## General Definition of 3NF, BCNF

- ❖ Can simplify the 3NF definition to remove the reference to partial dependencies/2NF
- ❖ R is in 3NF if for every FD  $X \rightarrow Y$ , either
  - X is a superkey or
  - Y is a prime attribute
- ❖ R is in BCNF if for every FD  $X \rightarrow Y$ , X is a superkey
  - R in BCNF  $\Rightarrow$  R is in 3NF

CS 178

53

## Normal Forms Compared

- ❖ BCNF is preferable, but sometimes in conflict with the goal of dependency preservation
  - It's strictly stronger than 3NF
- ❖ Let's see algorithms to obtain:
  - A BCNF lossless join decomposition
  - A 3NF lossless join, dependency preserving decomposition
    - Read this on your own from the textbook

CS 178

54

## BCNF Decomposition Algorithm

Input: Relation R (consisting of all attributes), set of functional dependencies F  
Output: BCNF schema *result*  
*result* := {R}  
while there is a schema  $R_i$  in *result* that is not in BCNF  
{  
    let  $A \rightarrow B$  be a FD that violates BCNF in relation  $R_i$   
  
    *result* := (*result* -  $R_i$ )  $\cup$   $\{(R_i - B), (A,B)\}$   
}

CS 178

55


## Example 1

- ❖ R = (C,T,H,R,S)
    - Course (C), Time (T), Hour (H), Room (R), Section (S), Grade (G)
- $$\begin{array}{ll} C \rightarrow T & CS \rightarrow G \\ HS \rightarrow R & HR \rightarrow C \\ HT \rightarrow R & \end{array}$$
- Key = {HS}  
Prime attributes = {H,S}

CS 178


56

## Testing for 3NF, BCNF




- ❖ Is the schema in BCNF ?
  - Check if there are non-BCNF dependencies
- ❖ Is the schema in 3NF ?
  - Check if there are non-3NF dependencies
    - Is there a dependency to non-prime attribute from something that is not a key ?

## Normalization Procedure: Summary



- ❖ Input= (Set of dependencies F, Set of attributes - single table schema)
- 1. Use attribute set closure algo to find (a) keys and (b) prime attributes
  - ❖ Prune the search using the various "tricks"
- 2. Test each FD in F to see if it satisfies 3NF/BCNF properties
- 3. Decompose into smaller relations using decomposition algorithm
- 4. If BCNF is not dependency preserving, then go with a 3NF decomposition

## Summary



- ❖ We can always decompose into 3NF and get:
  - Lossless join
  - Dependency preservation
- ❖ But with BCNF we are only guaranteed lossless joins
- ❖ BCNF is stronger than 3NF: every BCNF schema is also in 3NF
- ❖ The BCNF algorithm is nondeterministic, so there is not a unique decomposition for a given schema R

## So how do you design a schema from ground up ?



- ❖ Go over the application specifications
  - Can represent as ER diagram
- ❖ Identify all required information
  - This will constitute your "data" / attributes
- ❖ Identify the 'business rules'
  - This will define the functional dependencies
  - Will also define some of your application logic
  - This step may require you to "interact" with the "client" to clarify your questions
- ❖ Apply decomposition algorithm to get a good schema