

# CS460: Intro to Database Systems

## Class 19: *Functional Dependencies*

Instructor: Manos Athanassoulis

<https://bu-disc.github.io/CS460/>

# Review: Database Design

## Requirements Analysis

user needs; what must database do?

## Conceptual Design

high level description (often done w/ ER model)

## Logical Design

translate ER into DBMS data model

## Schema Refinement

consistency, normalization

## Physical Design

indexes, disk layout

# Review: Database Design

## Requirements Analysis

user needs; what must database do?

## Conceptual Design

high level description (often done w/ ER model)

## Logical Design

translate ER into DBMS data model

## Schema Refinement

**consistency, normalization**

## Physical Design

indexes, disk layout

# Why schema refinement

what is a bad schema?

*a schema with redundancy!*



why?



redundant storage & insert/update/delete anomalies

how to fix it?



***normalize*** the schema by decomposing  
normal forms: BCNF, 3NF, ... [next time]

# Motivating Example

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	Kurt	25K	617-555-3761

primary key?   
(SSN, Telephone)

problems of the schema? 

# Motivating Example

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	Kurt	25K	617-555-3761

## Problems

Storage

Update

Insert

Delete



# Motivating Example

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	Kurt	25K	617-555-3761

## Problems

**Storage:** store *Salary* multiple times

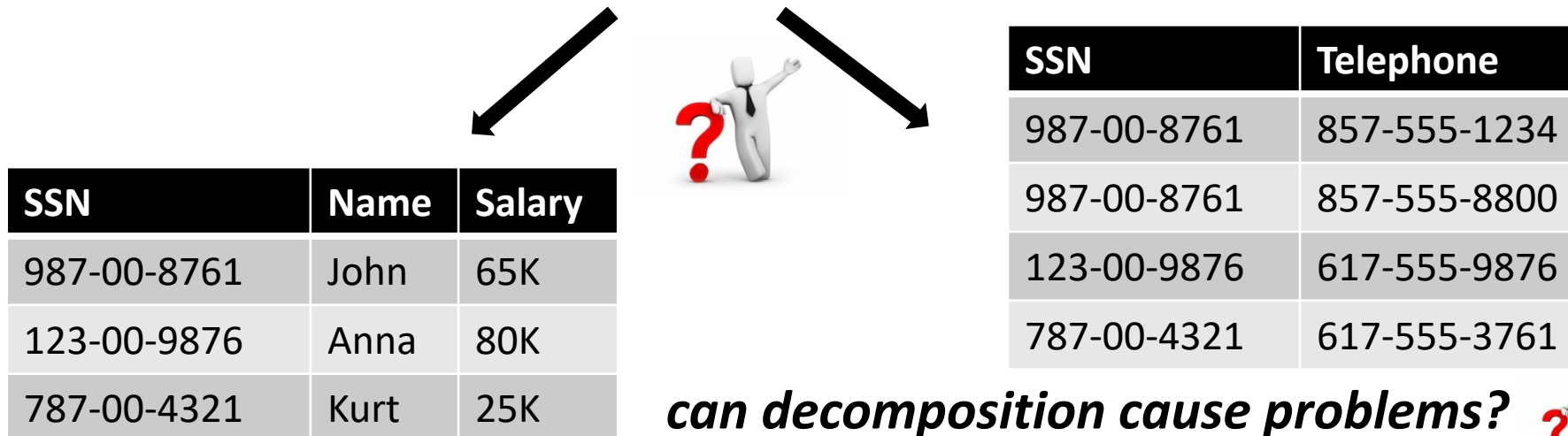
**Update:** change John's salary?

**Insert:** how to store someone with *no phone*?

**Delete:** how to delete Kurt's phone?

# Solution: Decomposition

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	Kurt	25K	617-555-3761



*can decomposition cause problems?* 

*how to find good decompositions?*



# FUNCTIONAL DEPENDENCIES

# Functional Dependencies

## Definition

Functional Dependencies (FDs) : form of constraint  
“generalized keys”

let  $X, Y$  nonempty sets of attributes of relation  $R$   
let  $t_1, t_2$  tuples :  $t_1.X = t_2.X$ , then  $t_1.Y = t_2.Y$

*“ $X \rightarrow Y$ ” : “ $X$  (functionally) determines  $Y$ ”*

*an FD comes from the application (not the data)*  
*an FD cannot be inferred (only validated)*

# Functional Dependencies

SSN	Name	Salary	Telephone
987-00-8761	John	65K	857-555-1234
987-00-8761	John	65K	857-555-8800
123-00-9876	Anna	80K	617-555-9876
787-00-4321	Kurt	25K	617-555-3761

which attribute determines which?



~~$SSN \rightarrow Telephone$~~

$SSN \rightarrow Name, Salary$

$SSN, Salary \rightarrow Name$

# FD: Example 3

studentID	classID	Semester	Instructor
1234	15	2	Mark
0043	15	1	Evimaria
4322	15	2	Mark
9876	175	4	Dora
1211	177	4	Manos
0043	154	2	Abraham

which attribute determines which?

*classID, Semester  $\rightarrow$  Instructor*

~~*studentID  $\rightarrow$  Semester*~~

*studentID, classID  $\rightarrow$  Semester*



# Reasoning about FDs

an FD holds for **all** allowable relations (**legal**)  
identified based on semantics of application

given an instance  $r$  of  $R$  and an FD  $f$ :

- (1) we can check whether  $r$  **violates**  $f$
- (2) we **cannot** determine if  $f$  **holds**

“ $K \rightarrow \text{all attributes of } R$ ” then  $K$  is a **superkey** for  $R$   
(does not require  $K$  to be *minimal*)

remember: in order to be a **candidate key** minimality is required

FDs are a generalization of keys

# Reasoning about FDs (Splitting)

assume  $A, B \rightarrow C, D$

$C, D$  are *independently determined* by  $A, B$   
so, we can split:  $A, B \rightarrow C$  and  $A, B \rightarrow D$

it does not work vice versa  
we cannot infer:  $A \rightarrow C, D$  or  $B \rightarrow C, D$

# Trivial FDs

for every relation

$$A \rightarrow A$$

$$A, B, C \rightarrow A$$

these are not informative!

in general an FD  $X \rightarrow A$  is called trivial if  $A \subseteq X$

it always holds!

# Identifying FDs

FD comes from the application (domain)

property of app semantics (not of instance)

cannot infer from an instance

given a set of tuples (instance  $r$ ), we can:

(1) confirm that an FD **might be** valid

(2) infer that an FD is **definitely invalid**

*but we cannot prove that an FD is valid*



# FD: Example 3

name	category	color	price	department
iPhone	smartphone	black	600	phones
Lenovo Yoga	laptop	grey	800	computers
unifi	networking	white	150	computers
unifi	cables	white	10	stationary
OnePlus	smartphone	silver	450	phones

~~name  $\rightarrow$  department~~



name, category  $\rightarrow$  department  
*we do not know!*



# Why use FDs?

the capture (and generalize) key constraints

offer more integrity constraints

help us detect redundancies

tell us how to normalize

*it is the principled way to solve the redundancy problem*

# More on: Reasoning for FD

when a set of FD holds over a relation

more FD can be inferred

**Armstrong's Axioms**

# Axiom 1: Reflexivity

for every subset  $X \subseteq \{A_1, \dots, A_n\}$

$$A_1, \dots, A_n \rightarrow X$$

## Examples

$$A, B \rightarrow B$$

$$A, B, C \rightarrow B, C$$

$$A, B, C \rightarrow A, B, C$$

# Axiom 2: Augmentation

for any attribute sets  $X, Y, Z$   
if  $X \rightarrow Y$ , then  $X, Z \rightarrow Y, Z$

## Examples

$A \rightarrow B$  then  $A, C \rightarrow B, C$

$A, B \rightarrow C$  then  $A, B, C \rightarrow C$

(here  $X=A,B$  and  $Y=Z=C$ )

# Axiom 3: Transitivity

for any attribute sets  $X, Y, Z$   
if  $X \rightarrow Y$  and  $Y \rightarrow Z$  then  $X \rightarrow Z$

## Examples

$A \rightarrow B$  and  $B \rightarrow C$  then  $A \rightarrow C$

$A \rightarrow B, C$  and  $B, C \rightarrow D$  then  $A \rightarrow D$

# Union and Decomposition

**rules that follow from AA**

*Union*

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

*Decomposition*

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

# Applying AA

Product

name	category	color	price	department
------	----------	-------	-------	------------

we know:

- (1)  $\text{name} \rightarrow \text{color}$
- (2)  $\text{category} \rightarrow \text{department}$
- (3)  $\text{color, category} \rightarrow \text{price}$

can we infer:  $\text{name, category} \rightarrow \text{price}$  

- (i) augmentation to (1):
  - (4)  $\text{name, category} \rightarrow \text{color, category}$
- (ii) transitivity to (4), (3)
  - $\text{name, category} \rightarrow \text{price}$



# Applying AA

Product

name	category	color	price	department
------	----------	-------	-------	------------

we know:

- (1)  $\text{name} \rightarrow \text{color}$
- (2)  $\text{category} \rightarrow \text{department}$
- (3)  $\text{color, category} \rightarrow \text{price}$

can we infer:  $\text{name, category} \rightarrow \text{color}$  

- (i) by reflexivity:
  - (5)  $\text{name, category} \rightarrow \text{name}$
- (ii) transitivity to (5), (1)
  - $\text{name, category} \rightarrow \text{color}$

# FD Closure

how can we find all FD?

## FD Closure

if  $F$  is a set of FD, the closure  $F^+$  is the set of all FDs logically implied by  $F$

*Using Armstrong Axioms we can find  $F^+$*

**sound:** any generated FD belongs to  $F^+$

**complete:** repeated application of AA generates  $F^+$

# Attribute Closure

$X$  an attribute set, the closure  $X^+$  is  
the set of all attributes  $B : X \rightarrow B$

in other words :  
attribute closure of  $X$  is the set of all attributes that  
“are (functionally) determined by  $X$ ”

# Applying AA

Product

name	category	color	price	department
------	----------	-------	-------	------------

we know:

- (1)  $\text{name} \rightarrow \text{color}$
- (2)  $\text{category} \rightarrow \text{department}$
- (3)  $\text{color, category} \rightarrow \text{price}$

Attribute closure: 

- (i) Closure of name  
 $\{\text{name}\}^+ = \{\text{name, color}\}$
- (i) Closure of name, category  
 $\{\text{name, category}\}^+ = \{\text{name, color, category, department, price}\}$

# Calculating Attribute Closure

let  $X = \{A_1, \dots, A_n\}$

$closure = X$

**UNTIL**  $closure$  does not change **REPEAT**:

**IF**  $B_1, \dots, B_m \rightarrow C$  **AND**

$B_1, \dots, B_m$  are all in  $closure$

**THEN** add  $C$  to  $closure$

# Calculating Attribute Closure

Example:  $R(A,B,C,D,E,F)$

$A, B \rightarrow C$

$A, D \rightarrow E$

$B \rightarrow D$

$A, F \rightarrow B$

$\{A,B\}^+$

$\{A,F\}^+$

# Calculating Attribute Closure

Example:  $R(A,B,C,D,E,F)$      $\{A,B\}$

<del><math>A, B \rightarrow C</math></del>	$\{A,B,C\}$
<del><math>A, D \rightarrow E</math></del>	$\{A,B,C,D\}$
<del><math>B \rightarrow D</math></del>	$\{A,B,C,D,E\}$
$A, F \rightarrow B$	

$\{A,B\}^+$

$\{A,F\}^+$



# Calculating Attribute Closure

Example:  $R(A,B,C,D,E,F)$      $\{A,B\}$

~~$A, B \rightarrow C$~~      $\{A,B,C\}$

~~$A, D \rightarrow E$~~      $\{A,B,C,D\}$

~~$B \rightarrow D$~~      $\{A,B,C,D,E\}$

~~$A, F \rightarrow B$~~

$\{A,B\}^+$

$\{A,F\}^+$



$\{A,F\}$

$\{A,F,B\}$

$\{A,F,B,C\}$

$\{A,F,B,C,D\}$

$\{A,F,B,C,D,E\}$



# Calculating Attribute Closure

Example:  $R(A,B,C,D,E,F)$

$A, B \rightarrow C$

$A, D \rightarrow E$

$B \rightarrow D$

$A, F \rightarrow B$

$\{A,B\}^+ = \{A,B,C,D,E\}$

$\{A,F\}^+ = \{A,F,B,C,D,E\}$

# Why calculate attribute closure?

for “does  $X \rightarrow Y$  hold” questions check if  $Y \subseteq X^+$

to compute the closure  $F^+$  of FDs

- (i) for each subset of attributes  $X$ , compute  $X^+$
- (ii) for each subset of attributes  $Y \subseteq X^+$ , output the FD  $X \rightarrow Y$

*why do we need the FD closure?  
to decide on decomposition (next time)*

# FD and Keys

*in terms of relational model*

superkey: a set of attributes such that:  
no two distinct tuples can have same values in all key fields

*in terms of FD*

superkey: a set of attributes  $A_1, A_2, \dots, A_n$  such that  
for any attribute  $B$ :  $A_1, A_2, \dots, A_n \rightarrow B$

key (or candidate key): requires minimality  
what if we have multiple candidate keys?  
- we specify one to be the **primary key**



# Computing (Super)Keys

(1) compute  $X^+$  for all sets of attributes  $X$

(2) if  $X^+ = \text{all attributes}$ , then  $X$  is a *superkey*  
*why?*



- because then " $X$  determines `all attributes`"

(3) if, also, *no subset of  $X$  is superkey*  
then  $X$  is also a key

# Example

Product

name	category	color	price
------	----------	-------	-------

we know:

- (1)  $\text{name} \rightarrow \text{color}$
- (2)  $\text{color, category} \rightarrow \text{price}$

Superkeys:

$\{\text{name, category}\}$ ,  $\{\text{name, category, price}\}$ ,  
 $\{\text{name, category, color}\}$ ,  $\{\text{name, category, price, color}\}$

Keys:

$\{\text{name, category}\}$



# Can we have more than 1 key?



what about the relation ***R*** (*A,B,C*) with:

$A, B \rightarrow C$

$A, C \rightarrow B$

which are the keys?

$\{A, B\}$  and  $\{A, C\}$  are both minimal

# Should we use all FDs?

given a set of FDs  $F$  we have discussed about  $F^+$

the useful info is in the minimal cover of  $F$

“the smallest subset of FDs  $S$ :  $S^+ = F^+$ ”

**Formally:** minimal cover  $S$  for a set of FDs  $F$ :

- (1)  $S^+ = F^+$
- (2) RHS of each FD in  $S$  is a single attribute
- (3) if we remove any FD from  $S$  or remove any attribute from its LHS the closure is not  $F^+$

# Example of Minimal Cover

$R(C, S, J, D, P, Q, V)$

key  $C$  ( $C^+ = \{C, S, J, D, P, Q, V\}$ )

$J, P \rightarrow C$

$S, D \rightarrow P$

$J \rightarrow S$

Minimal cover:

$C \rightarrow J, C \rightarrow D, C \rightarrow Q, C \rightarrow V$

$J, P \rightarrow C$

$S, D \rightarrow P$

$J \rightarrow S$

This is useful to decide how to solve the problem of redundancy (decomposition)!

More on that next time!!



# Summary

## Functional Dependencies and (Super)Keys

Reasoning with FDs:

- (1) given a set of FDs, infer all implied FDs
- (2) given a set of attributes  $X$ , infer all attributes that are functionally determined by  $X$

*Next: how to use detect that a table is “bad”?*