DSAA 5012
Advanced Data Management for Data Science

LECTURE 9
RELATIONAL DATABASE DESIGN: FUNCTIONAL DEPENDENCIES

RELATIONAL DATABASE DESIGN: OUTLINE

Functional Dependencies
- Definition
- Functional Dependencies and Keys
- Inference Rules for Functional Dependencies
- Closure of Attribute Sets
- Canonical Cover

Normalization
- Goals
- First Normal Form (1NF)
- Second Normal Form (2NF)
- Third Normal Form (3NF)
- Boyce-Codd Normal Form (BCNF)

RELATIONAL DATABASE DESIGN AND FUNCTIONAL DEPENDENCIES

- Relational database design requires that we find a "good" collection of relation schemas.

  ☞ **A bad design may lead to several problems.**

- Functional dependencies can be used to refine a relation schema reduced from an E-R schema by iteratively decomposing it (called normalization) to place it in a certain normal form.
  - The first four normal forms $\Rightarrow$ use only functional dependencies.
  - Additional normal forms $\Rightarrow$ use other types of dependencies

☞ **Normal forms do not guarantee a good design!**

FUNCTIONAL DEPENDENCY (FD): DEFINITION

Let R be a relation schema, X, Y be sets of attributes in R and *f* be a time-varying function from X to Y. Then

$$f\!: X \rightarrow Y$$

is a *functional dependency (FD)* if *at every point in time*, for a given value of *x* in X there is at most one value of *y* in Y.

Example: PGStudent(studentId, name, supervisor, specialization)

$$f\!: \text{supervisor} \rightarrow \text{specialization}$$

Consequences of the FD:

- If two student records have the same supervisor (e.g., Papadias), then they must have the same specialization (e.g., Database).
- On the other hand, if two student records have different supervisors, then they may have the same or different specializations.

PGStudent

studentId	name	supervisor	specialization
23455789	Bruno Ho	Yang	Artificial Intelligence
23556789	Jenny Jones	Papadias	Database
25678989	Kathy Ko	Kim	Software Technology
26789012	Susan Sze	Papadias	Database
26184624	Terry Tam	Song	Artificial Intelligence
26186666	Carol Chen	Tai	Graphics

FUNCTIONAL DEPENDENCY (FD): DEFINITION

- We normally omit the "*f*:" and simply write the FD as $X \rightarrow Y$.
  - X is called the determinant set or left-hand side (LHS) of the FD.
  - Y is called the dependent set or right-hand side (RHS) of the FD.

  ☞ **We say that X determines Y or Y depends on X.**

- A functional dependency $X \rightarrow Y$ is trivial if $Y \subseteq X$.

  ☞ **Y appears on both the LHS and the RHS of the FD.**

  - Trivial FDs hold for all relation instances.

- A functional dependency $X \rightarrow Y$ is non-trivial if $Y \cap X = \varnothing$.

  ☞ **Y does not appear on both the LHS and the RHS of the FD.**

  - Non-trivial FDs are given as constraints when designing a database.
  - Non-trivial FDs constrain the set of legal relation instances.

- In general, X (and Y) can be sets of attributes.

  E.g., if $X \equiv X_1 \cup X_2 \cup \ldots \cup X_n$, then we can write $X_1X_2\ldots X_n \rightarrow Y$

RELATIONAL DATABASE DESIGN: FUNCTIONAL DEPENDENCIES

EXERCISES 1, 2

EXERCISE 1

Assume that this table contains the *only* set of tuples that may appear in a relation R(X, Y, V, W). Which of the following FDs hold in R?

tuple	X	Y	V	W
1	x_1	y_1	v_1	w_1
2	x_1	y_1	v_2	w_2
3	x_2	y_1	v_1	w_3
4	x_2	y_1	v_3	w_4

X→X	**Yes**	– trivial (holds in any relation)
X→Y	**Yes**	– for a given X value all Y values are identical
X→V	**No**	– V values differ for same X value (e.g., tuples 1 & 2)
X→W	**No**	– W values differ for same X value (e.g., tuples 1 & 2)
Y→X	**No**	– X values differ for same Y value (e.g., tuples 2 & 3)
W→X	**Yes**	– all W values are different
XV→Y	**Yes**	– X alone determines Y
YV→X	**No**	– X values differ for same YV value in tuples 1 and 3

EXERCISE 2

In Exercise 1, we assumed that we know all possible records in the table, which is not usually true. In general, by looking at an instance of a relation, we can only tell FDs that are not satisfied.

List 5 FDs that are not satisfied in the table.

A	B	C
a_1	b_1	c_1
a_1	b_1	c_2
a_2	b_1	c_1
a_2	b_1	c_3

$A \rightarrow C$

$B \rightarrow A$

$B \rightarrow C$

$C \rightarrow A$

$AB \rightarrow C$

FUNCTIONAL DEPENDENCIES AND KEYS

- An FD is a generalization of the concept of a *key.*

  For the relation

  PGStudent (studentId, name, supervisor, specialization)

  we can write:

  studentId → name, supervisor, specialization

  because the key, studentId, determines the (single) value of all the attributes (i.e., the entire tuple).

- If two tuples in the PGStudent relation have the same studentId, then they must have the same values on all attributes.

☞ **They must be the same tuple!**
(Since the relational model does not allow duplicate tuples.)

INFERENCE RULES FOR FDS

- Given a set of functional dependencies F, there are certain other functional dependencies that are *logically implied* by F.
- We can find all functional dependencies implied by F by applying the following inference rules for FDs.

IR1: Reflexivity	If $Y \subseteq X$, then $X \rightarrow Y$ (*trivial FD*)	**Armstrong's Axioms** **(basic rules)**
IR2: Augmentation	$X \rightarrow Y \vDash XZ \rightarrow YZ$	
IR3: Transitivity	$X \rightarrow Y,\ Y \rightarrow Z \vDash X \rightarrow Z$	
IR4: Union	$X \rightarrow Y,\ X \rightarrow Z \vDash X \rightarrow YZ$	**Additional rules** **(derivable from IR1, IR2 and IR3)**
IR5: Decomposition	$X \rightarrow YZ \vDash X \rightarrow Y$ and $X \rightarrow Z$	
IR6: Psuedotransitivity	$X \rightarrow Y,\ WY \rightarrow Z \vDash WX \rightarrow Z$	

EXAMPLES OF ARMSTRONG'S AXIOMS

IR1: Reflexivity If $Y \subseteq X$, then $X \rightarrow Y$ (*trivial FD*)

name → name	since name ⊆ name
name, supervisor → name	since name ⊆ {name, supervisor}
name, supervisor → supervisor	since supervisor ⊆ {name, supervisor}

IR2: Augmentation $X \rightarrow Y \vDash XZ \rightarrow YZ$

studentId → name	(given)
studentId, supervisor → name, supervisor	(inferred)

IR3: Transitivity $X \rightarrow Y, Y \rightarrow Z \vDash X \rightarrow Z$

studentId → supervisor	(given)
supervisor → specialization	(given)
studentId → specialization	(inferred)

INFERENCE RULES FOR FDS AND CLOSURE

Inference rules IR1, IR2 and IR3 are sound and complete.

sound: Given a set of FDs, F, specified on a relation schema R, any FD that we can infer from F by using **IR1** to **IR3** *will hold* in every relation instance *r* of R that satisfies F (i.e., it is a *true* FD).

complete: Using **IR1**, **IR2** and **IR3** repeatedly to infer FDs will infer *all* the FDs that can be inferred from F.
(i.e., there are *no other FDs* that are true).

The set of all functional dependencies *logically implied* by F is called the closure of F denoted as F^+.

EXAMPLE OF FDS IN THE CLOSURE F^+

Given: R(A, B, C, G, H, I)

F = {A→B
A→C
CG→H
CG→I
B→H}

Some members of F^+

A→H
AG→I
CG→HI

ATTRIBUTE CLOSURE

- The closure of X under F (denoted by X^+) is the set of attributes that are functionally determined by X under F.

$X \rightarrow Y$ is in $F^+ \Leftrightarrow Y \subseteq X^+$

X is a set of attributes

Given studentId (e.g., studentId is X).

If studentId $\rightarrow$ name is in F^+
then name is part of studentId$^+$
(i.e., studentId$^+$= {studentId, name, …})

Why is studentId in X$^+$?

If studentId $\rightarrow$ supervisor is in F^+
then supervisor is part of studentId$^+$
(i.e., studentId$^+$= {studentId, name, supervisor, …})

ATTRIBUTE CLOSURE: ALGORITHM

Input:

R a relation schema
F a set of functional dependencies
$X \subset R$ (the set of attributes for which we want to compute the closure)

Output:

X^+ the closure of X w.r.t. F

$X^{(0)} := X$
Repeat
$X^{(i+1)} := X^{(i)} \cup Z$, where Z is the set of attributes such that there exists
$Y \rightarrow Z$ in F, and $Y \subset X^{(i)}$
Until $X^{(i+1)} := X^{(i)}$
Return $X^{(i+1)}$

☞ **For every attribute Y in X^i, if Y is the LHS of an FD, then add the RHS attributes Z to the closure; repeat until there are no more attributes to add.**

ATTRIBUTE CLOSURE: USES

Testing for Superkey

To test if X is a superkey, compute X^{+}, and check if X^{+} contains all attributes of R. If X is minimal, then it is a candidate key.

☞ An attribute that is part of *any* candidate key is called a prime attribute; otherwise it is a non-prime attribute.

Testing Functional Dependencies

To determine whether a functional dependency $X \rightarrow Y$ holds (i.e., if $X \rightarrow Y$ is in F^{+}), compute X^{+} and check if $Y \subseteq X^{+}$.

Computing the Closure of *F*

For each subset $X \subseteq R$, compute X^{+} and, for each $Y \subseteq X^{+}$, output a functional dependency $X \rightarrow Y$.

ATTRIBUTE CLOSURE: EXAMPLE

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

F = { AB→C, C→A, BC→D, ACD→B,
D→EG, BE→C, CG→BD, CE→AG }

☞ Compute the closure of X = BD w.r.t F

$X^{(0)} = \{B, D\}$

$X^{(1)} = \{B, D, E, G\}$ apply D→EG add E, G to X

$X^{(2)} = \{B, C, D, E, G\}$ apply BE→C add C to X

$X^{(3)} = \{A, B, C, D, E, G\}$ apply C→A add A to X

$X^{(4)} = X^{(3)}$ no more FDs can be applied

$\{B, D\}^+ = \{A, B, C, D, E, G\}$ **Is BD a candidate key?**

RELATIONAL DATABASE DESIGN: FUNCTIONAL DEPENDENCIES

EXERCISES 3, 4

EXERCISE 3

Given relation schema R(X, Y, U, V, W) and F = {X→Y, UV→W, V→X}

a) Determine the closure of each attribute.

X^+ = {X, Y} (Look for X on LHS of FDs)

Y^+ = {Y}

U^+ = {U}

V^+ = {V, X, Y}

W^+ = {W}

b) What are the candidate keys of R?

The candidate key is UV since UV^+ = {X, Y, U, V, W}.

EXERCISE 4

Given relation schema R(A, B, C, G, H, I) and

F = {A→B, A→C, CG→H, CG→I, B→H}

a) Is AG a (super)key of R given F? **Yes** **Why?** Since AG→R

Compute AG^+

$AG^{(0)} = \{A, G\}$
$AG^{(1)} = \{A, G, B\}$ (A→B and $A \subseteq \{A, G\}$)
$AG^{(2)} = \{A, G, B, C\}$ (A→C and $A \subseteq \{A, G\}$)
$AG^{(3)} = \{A, G, B, C, H\}$ (CG→H and $CG \subseteq \{A, G, B, C\}$)
$AG^{(4)} = \{A, G, B, C, H, I\}$ (CG→I and $CG \subseteq \{A, G, B, C, H\}$)

b) Is AG a candidate key? **Yes** **Why?** Cannot remove A or G

c) Does $A^+ \rightarrow R$ hold? **No** since $A^+ = \{A, B, C, H\}$

d) Does $G^+ \rightarrow R$ hold? **No** since $G^+ = \{G\}$

REDUNDANCY OF FDS

- Sets of functional dependencies may have redundant functional dependencies that can be inferred from the others.

  $A \rightarrow C$ is redundant in: $\{A \rightarrow B, B \rightarrow C, A \rightarrow C\}$ **WHY?**

- Parts of a functional dependency may be redundant.

  Example of extraneous/redundant attribute in RHS.

  $\{ A \rightarrow B, B \rightarrow C, A \rightarrow CD \}$ can be simplified to
  $\{ A \rightarrow B, B \rightarrow C, A \rightarrow D \}$ **WHY?**

  Example of extraneous/redundant attribute in LHS (partial dependency).

  $\{ A \rightarrow B, B \rightarrow C, AC \rightarrow D \}$ can be simplified to
  $\{ A \rightarrow B, B \rightarrow C, A \rightarrow D \}$ **WHY?**

CANONICAL COVER

A *canonical cover* for F is a set of functional dependencies F_c such that

- F and F_c are equivalent (i.e., they have the same closure F^+).
- F_c contains no redundancy (i.e., no extraneous attributes).
- The LHS of each functional dependency in F_c is unique.

☞ **Every set of FDs has a canonical cover.**

☞ **The canonical cover for a set of FDs is not unique.**

☞ **Testing for FD violation using F_c is usually simpler.**

CANONICAL COVER: ALGORITHM

☞ **Recall that two FDs X→Y, X→Z, can be converted to X→YZ (IR4: Union).**

Algorithm to compute the canonical cover of F.

1. $F_c = F$
2. Repeat:
   a) Use the union rule (**IR4**) to replace any FDs in F_c of the form X→Y and X→Z with X→YZ.
   b) Find an FD X→Y in F_c with an extraneous attribute either in X or in Y.
      If an extraneous attribute is found, delete it from X→Y in F_c.
3. until F_c does not change.

☞ **The union rule may become applicable after some extraneous attributes have been deleted, so it must be reapplied.**

CANONICAL COVER: EXAMPLE COMPUTATION

R(A, B, C)

F = {A→BC, B→C, A→B, AB→C}

- Use union rule (**IR4**) to combine A→BC and A→B into A→BC
  - F_c is now {A→BC, B→C, AB→C}
- Delete extraneous attribute A in AB→C because of B→C.
  - F_c is now {A→BC, B→C}
- Delete extraneous attribute C in A→BC because of A→B and B→C.
  - F_c is now {A→B, B→C}

☞ The canonical cover F_c = {A→B, B→C}

RELATIONAL DATABASE DESIGN: FUNCTIONAL DEPENDENCIES

EXERCISES 5, 6, 7

Upload your completed exercise worksheet to Canvas by **11 p.m. on March 3rd.**