Functional Dependencies

SWEN304/SWEN439

Lecturer: Dr Hui Ma

Engineering and Computer Science

TE WHARE WĀNANGA O TE ŪPOKO O TE IKA A MĀUI
VICTORIA
UNIVERSITY OF WELLINGTON

Outline

- Definition of functional dependency
- Semantics of a functional dependency
- Closure of a set of functional dependencies
- Finding a "minimal" cover
- Functional dependencies and a relation schema key
- *Readings from the textbook:*
  - *Chapter 15*

Functional Dependency

- One of the most important constraints for the relational database design
- Let $URS(U, C)$ be given, and $X, Y \subseteq U$
- The **functional dependency** (abbreviated FD) between attribute sets X and Y is an expression of the form

$$f : X \rightarrow Y,$$

  where f is an (optional) name, X is left-hand side $LHS(f)$, and Y is right-hand side $RHS(f)$
- X functionally defines Y, and Y functionally depends on X

Semantics of a Functional Dependency

- The meaning of the expression

  $$f: X \rightarrow Y$$

  is that with each particular X value there is always the same Y value associated
- A functional dependency is a semantic constraint that can be defined only by considering rules of behavior in the UoD
- A functional dependency $X \rightarrow Y$ is to be defined only when it is known that in the real world, each X value is associated with at most one Y value

Functional Dependency - Notations

- An expression having semantically defined attributes:

$$\{StudId, CourId\} \rightarrow \{Grade, Year\}$$

will be considered as being equivalent to

$$StudId + CourId \rightarrow Grade + Year$$

- If the sets are singletons, then

$$StudId \rightarrow SName$$

- An expression having semantically undefined attributes

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

will be considered as being equivalent to

$$AB \rightarrow CD, \text{ or } A + B \rightarrow C + D$$

FDs Satisfied by the Relation "Faculty"

StId	*StName*	*NoPts*	*CourId*	*CoName*	*Grd*	*LecId*	*LeName*
007	James	80	M114	Math	A+	*777*	Mark
131	Susan	18	C102	Java	B-	*101*	Ewan
007	James	80	C102	Java	A	*101*	Ewan
555	Susan	18	M114	Math	B+	*999*	Vladimir
007	James	80	C103	Algorith	A+	*99*	Peter
131	Susan	18	M214	Math	ω	*333*	Peter
555	Susan	18	C201	C++	ω	*222*	Robert
007	James	80	C201	C++	A+	*222*	Robert
010	John	0	C101	Inet	ω	*820*	Ray

Defining Functional Dependencies

- UOD_1
  - Consider the set of attributes

    $\{StudId, CourId, Grade\}$
  - and the rule of behavior

    "A student can enroll a course at most once".
  - Then

$$StudId + CourId \rightarrow Grade$$

Defining Functional Dependencies

- UOD_2
  - Consider the set of attributes {*StudId*, *CourId*, *Term*, *Grade*}, and the rule of behavior “A student can enroll a course more than once, but each time in a different term”. Then

$$StudId + CourId + Term \to Grade$$

  - Consider the set of attributes {*StudId*, *CourId*, *Term*, *AssigNo, Marks*}, and the rule of behavior “A student can enroll a course more than once, but each time in a different term and each time can do each assignment only once”. Then

$$StudId + CourId + Term + AssigNo \to Marks$$

Recall: The Implication Operation from Logic

- Implication $p \Rightarrow q$ is a logic operation
  - q is a logical consequence of p
- $p \Rightarrow q$ is true if either the antecedent (p) is false or the consequent (q) is true
- Recall: The truth table of the implication operation

p	q	$\Rightarrow$
False	False	
False	True	
True	False	
True	True	

- We will use it several times in our lectures
  - For example, for the definition of functional dependency

Satisfaction of a Functional Dependency

- A **particular** relation $r(U)$ satisfies the functional dependency $X \rightarrow Y$ if

  $$(\forall u, v \in r(U))(u[X] = v[X] \Rightarrow u[Y] = v[Y])$$

  - i.e., whenever two tuples agree on all attributes in X, they also agree on all attributes in Y

- **Note:** This statement considers only one particular relation
  - To claim that FD $X \rightarrow Y$ is generally valid, we would have to consider all relations over (U, C) that are plausible in the perceived UoD
  - The set of all FDs F that are valid in the UoD is a subset of C - the set of relation schema constraints C

Some Questions

Department

LecId	*LeName*	*CourId*	*CoName*	*DptId*	*DptName*
12	Ewan	C102	Java	CS	Comp Sc
33	Pavle	C302	DB Sys	CS	Comp Sc

- Does this particular *Department* relation satisfy the functional dependency $LecId \rightarrow CourId$?
- Is $LecId \rightarrow CourId$ valid in the UoD?
  - Can we conclude that in the CS Department each lecturer always teaches at most one paper?

- Does this particular *Department* relation satisfy the functional dependency $DptId \rightarrow CourId$?
- Is $DptId \rightarrow CourId$ valid in the UoD?

Redundant Functional Dependencies

- A given set of functional dependencies can contain some redundant ones

- Redundant functional dependencies are those that are a logical consequence of some other ones, or that are trivial

- FD on *URS* is said to be trivial if it is satisfied by all relations over (U, C)
  - an FD $X \rightarrow Y$ on *URS* is trivial if and only if $Y \subseteq X$ holds

Redundant Functional Dependency Examples

- Suppose the following set of FDS is given

  $F = \{StdId \rightarrow StName,\ CourId \rightarrow CoName\ ,\ LecId \rightarrow LecName\ ,\ LecId \rightarrow CourId\}$

- Redundant FDs:
  - $StName \rightarrow StName$

    (trivial),
  - $CourId + StdId \rightarrow CoName$

    (redundant – consequence of $CourId \rightarrow CoName$),
  - $LecId +\ LecName \rightarrow CourID$

    (redundant – consequence of $LecId \rightarrow CourId$),
  - $LecId \rightarrow CoName$

    (transitive – consequence of $LecId \rightarrow CourId$ and $CourId \rightarrow CoName$)

Redundant Functional Dependencies

- Functional dependencies are constraints that, as all other constraints, when once defined, should be satisfied in a database
- Redundant functional dependencies are satisfied when the basic ones are satisfied
- Accordingly, redundant FDs are noxious, because their satisfaction checking is just using precious computer resources in vain

Covers of a Set of FDs

- The goal is to replace a given, potentially redundant, set of FDs *F* with another one *E* that contains only functional dependencies that are necessary and sufficient to describe perceived rules of UoD behavior
- That replacement may be done only if each FD in *F* is either contained in *E* or represents a logical consequence of *E*
- A set of functional dependencies *E* is said to cover another set of functional dependencies *F* if every FD in *F* is also in E^+
- *F* and *E* are said to be equivalent, or to have equal closures ($F^+ = E^+$), also it is said that they cover each other

Closure of a Set of FDs

- The closure of F (denoted F^+) contains all FDs in F and all consequences of F
- It is computed by an exhaustive application of **inference rules** on a given set F of FDs

Inference Rules

- Given U, F, and X, Y, Z, $W \subseteq U$

1. (Reflexivity) $Y \subseteq X \vDash X \to Y$ (trivial FD)
2. (Augmentation) $X \to Y \wedge W \subseteq Z \vDash XZ \to YW$ (partial FD)
3. (Transitivity) $X \to Y \wedge Y \to Z \vDash X \to Z$ (transitive FD)
4. (Decomposition) $X \to YZ \vDash X \to Y \wedge X \to Z$
5. (Union) $X \to Y \wedge X \to Z \vDash X \to YZ$
6. (Pseudo transitivity) $X \to Y \wedge WY \to Z \vDash WX \to Z$

   (if $W = \emptyset$, pseudo transitivity turns into transitivity)

- Inference rules 1, 2 and 3 are known as **Armstrong's inference rules**

Computing Closures

- One way to check whether one set of FDs can be replaced by another, is to check whether they have equal closures
- But computing the closure of a set F of FDs is very complex
- $|F^+| \geq 2^{|U|}$ (U is the universal set)
- Instead of comparing many sets of FDs and computing their closures, we look for a minimal cover of F directly
- This is done using the closure of a set of attributes

Closure of a Set of Attributes

- Given U, F and $X \subseteq U$
- Closure of X with regard to F is defined as

$$X_F^+ = \{A \in U \mid X \rightarrow A \in F^+\}$$

and is used in finding the minimal cover of F

Computing Closure of X (set of attributes)

```
X⁺ = X;                  // according to reflexivity
oldX⁺ = ∅
while (oldX⁺ ⊂ X⁺) {
    oldX⁺ = X⁺
    for (each FD Y → Z ∈ F) {
        if (Y ⊆ X⁺) {
        X⁺ = X⁺ ∪ Z;  //according to
                      // augmentation & transitivity
        }
    }
}
```

Example

```
X+= X;                    // according to reflexivity
    oldX+ = ∅
    while (oldX+ ⊂ X+) {
        oldX+ = X+
        for (each FD Y→Z ∈ F) {
            if (Y ⊆ X+) {
            X+ = X+ ∪ Z;  //according to
// augmentation & transitivity
            }
        }
    }
```

- $F = \{B \rightarrow C, A \rightarrow B\}$
- $A^+ =$

Minimal Cover

- A set of FDs G is a **minimal cover** of the set F if
  Each FD in G has a single attribute on its right hand side

1. G is left reduced (no one FD in G has any superfluous attribute on its left hand side, (a left reduced FD = total FD, a not reduced FD = partial FD))

$$(\forall X \to A \in G)(\forall B \in X)((X - B) \to A \notin G^+)$$

2. G is non-redundant (doesn't contain any trivial or pseudo transitive FD)

$$(\forall X \to A \in G)((G - \{X \to A\})^+ \subset G^+),$$

3. $F^+ = G^+$

Finding a Minimal Cover

1. Set $G = F$
2. Replace each FD $X \rightarrow \{A_1, A_2, ..., A_n\}$ in G with the following n FDs $X \rightarrow A_1$, $X \rightarrow A_2, ..., X \rightarrow A_n$
3. Do left reduction

   for each FD $X \rightarrow A$ in G do

   for each B in X do

   if $A \in (X - B)^+_G$ then

   $G = (G - \{X \rightarrow A\}) \cup \{(X - B) \rightarrow A\}$
4. Eliminate redundant FDs

   for each FD $X \rightarrow A$ in G do

   if $A \in (X)^+_{G - \{X \rightarrow A\}}$ then $G = G - \{X \rightarrow A\}$

Finding a Minimal Cover – Step 2

$$F = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow CD,\ AB \rightarrow C\}$$

- Apply the Decomposition Inference Rule

$$G = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow CD,\ AB \rightarrow C\}$$

- The Decomposition Inference Rule should be applied only onto functional dependencies having more than one attribute on their *RHS*

$$G_1 = \{A \rightarrow B,\ B \rightarrow C,\ A \rightarrow C,\ A \rightarrow D,\ AB \rightarrow C\}$$

Finding a Minimal Cover – Step 3

- Do Left Reduction
  - Only the functional dependencies having more than one attribute on their LHS may be reduced

    $$G_1 = \{A \to B,\ B \to C,\ A \to C,\ A \to D,\ \boxed{AB \to C}\ \}$$

  - To test whether there is a superfluous attribute on the *LHS*, we try to remove each of the *LHS* attributes and apply attribute closure algorithm to see if the *RHS* still functionally depends on the remainder of the *LHS*

$$(AB - A)^+ = B^+ = B \Rightarrow C \notin (AB - A)^+$$

$$(AB - B)^+ = A^+ = ABCD \Rightarrow C \in (AB - B)^+ \Rightarrow$$

$$G_2 = \{A \to B,\ B \to C,\ A \to C,\ A \to D,\ \cancel{A \to C}\ \}$$

(G_2 should contain only one $A \to C$)

Finding a Minimal Cover – Step 4

- Eliminate Redundant FDs
  - In principle, this step should be applied on each FD, but we shall consider only the highlighted one

$$H = \{A \to B,\ B \to C,\ A \to D,\ A \to C\}$$

  - To check whether a FD is redundant, we compute the attribute closure of its LHS with regard to the given set of FDs without the FD considered
  - If the RHS is in the attribute closure, then the FD is redundant

$$A^{+}_{H - \{A \to C\}} = ABCD \ \Rightarrow\ C \in A^{+}_{H - \{A \to C\}}$$

$$\Rightarrow H_1 = \{A \to B,\ B \to C,\ A \to D\}$$

FDs and a Relation Schema Key

- Each relation schema **key** is the consequence of a functional dependency from F^+
- Let $R(A_1, \ldots, A_n)$ be a relation schema and F the set of functional dependencies in R
- Set of attributes $X \subseteq R$ is a relation schema **key** if

$1^o \; X \rightarrow R \in F^+$

$2^o \; (\forall Y \subset X)(Y \rightarrow R \notin F^+)$

- Not null condition still applies to X
- A **prime** attribute is a relation schema attribute that belongs to any of the keys
- Primary key is still just one of the keys

A Key Finding Algorithm

$X := R$ (*X is initialized as a super key*)

for each A in X do

  if $R \subseteq (X - A)^+_F$ then

  $X := X - A$

- Example.
  - Given: $R = \{A, B, C\}$, $F = \{A \rightarrow B, B \rightarrow C\}$
  - $X = ABC$ (ABC is a superkey)
  - $(X - A)^+_F = BC$ (*So, our superkey is still $X = ABC$*)
  - $(X - B)^+_F = ABC$ (* B is not needed, so $X = AC$*)
  - $(X - C)^+_F = ABC$ (* C is not needed, so $X = A$ *)
  - $K(R) = A$

Summary

- The functional dependency is a semantic constraint that mirrors certain type of UoD rules of behavior
- Functional dependencies are important relational constraints
- Removing harmful redundant functional dependencies is done by finding a 'minimal' cover
- A minimal cover is found using the cover of a set of attributes as a tool
- A relation schema key is a consequence of a functional dependency
- Each attribute of a relational schema is functionally dependent on each of the keys