IF3111 Basis Data – Functional Dependency

Wikan Danar
Departemen Teknik Informatika
Institut Teknologi Bandung

Functional Dependencies

- Constraints on the set of legal relations.
- Require that the value for a certain set of attributes determines uniquely the value for another set of attributes.
- A functional dependency is a generalization of the notion of a *key.*

Functional Dependencies (Cont.)

- Let R be a relation schema

$$\alpha \subseteq R \;\; and \;\; b \subseteq R$$

- The functional dependency

$$\alpha \rightarrow b$$

holds on R if and only if for any legal relations r(R), whenever any two tuples t_1 and t_2 of r agree on the attributes α, they also agree on the attributes b. That is,

$$t_1[\alpha] = t_2[\alpha] \;\; \Rightarrow \;\; t_1[b] = t_2[b]$$

- Example: Consider *r(A,B)* with the following instance of *r.*

1	4
1	5
3	7

- On this instance, $A \rightarrow B$ does **NOT** hold, but $B \rightarrow A$ does hold.

Functional Dependencies (Cont.)

- K is a superkey for relation schema R if and only if $K \to R$
- K is a candidate key for R if and only if
  - $K \to R$, and
  - for no $\alpha \subset K$, $\alpha \to R$
- Functional dependencies allow us to express constraints that cannot be expressed using superkeys. Consider the schema:

  Loan-info-schema = (customer-name, loan-number, branch-name, amount).

  We expect this set of functional dependencies to hold:

  loan-number $\to$ *amount*
  loan-number $\to$ *branch-name*

  but would not expect the following to hold:

  loan-number $\to$ *customer-name*

Use of Functional Dependencies

- We use functional dependencies to:
  - test relations to see if they are legal under a given set of functional dependencies.
    - If a relation *r* is legal under a set *F* of functional dependencies, we say that *r* satisfies *F.*
  - specify constraints on the set of legal relations
    - We say that *F* holds on *R* if all legal relations on *R* satisfy the set of functional dependencies *F.*
- Note: A specific instance of a relation schema may satisfy a functional dependency even if the functional dependency does not hold on all legal instances.
  - For example, a specific instance of *Loan-schema* may, by chance, satisfy

    loan-number $\rightarrow$ *customer-name.*

Functional Dependencies (Cont.)

- *A* functional dependency is trivial if it is satisfied by all instances of a relation
  - *E.g.*
    - *customer-name, loan-number* $\rightarrow$ *customer-name*
    - *customer-name* $\rightarrow$ *customer-name*
  - In general, $\alpha \rightarrow b$ is trivial if $b \subseteq \alpha$

Closure of a Set of Functional Dependencies

- Given a set *F* set of functional dependencies, there are certain other functional dependencies that are logically implied by *F*.
  - E.g. If $A \rightarrow B$ and $B \rightarrow C$, then we can infer that $A \rightarrow C$
- The set of all functional dependencies logically implied by *F* is the *closure* of *F*.
- We denote the *closure* of *F* by F^+.
- We can find all of F^+ by applying Armstrong's Axioms:
  - if $b \subseteq \alpha$, then $\alpha \rightarrow b$ **(reflexivity)**
  - if $\alpha \rightarrow b$, then $\gamma\,\alpha \rightarrow \gamma\, b$ **(augmentation)**
  - if $\alpha \rightarrow b$, and $b \rightarrow \gamma$, then $\alpha \rightarrow \gamma$ **(transitivity)**
- These rules are
  - sound (generate only functional dependencies that actually hold) and
  - complete (generate all functional dependencies that hold).

Example

- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- some members of F^+
  - $A \rightarrow H$
    - by transitivity from $A \rightarrow B$ *and* $B \rightarrow H$
  - $AG \rightarrow I$
    - by augmenting $A \rightarrow C$ with G, to get $AG \rightarrow CG$
      and then transitivity with $CG \rightarrow I$
  - $CG \rightarrow HI$
    - from $CG \rightarrow H$ *and* $CG \rightarrow I$: "union rule" can be inferred from
      - definition of functional dependencies, or
      - Augmentation of $CG \rightarrow I$ to infer $CG \rightarrow CGI$, augmentation of $CG \rightarrow H$ to infer $CGI \rightarrow HI$, and then transitivity

Procedure for Computing F^+

- To compute the closure of a set of functional dependencies F:

$F^+ = F$
repeat
  for each functional dependency f in F^+
    apply reflexivity and augmentation rules on f
    add the resulting functional dependencies to F^+
  for each pair of functional dependencies f_1 and f_2 in F^+
    if f_1 and f_2 can be combined using transitivity
      then add the resulting functional dependency to F^+
until F^+ does not change any further

NOTE: We will see an alternative procedure for this task later

Closure of Functional Dependencies (Cont.)

- We can further simplify manual computation of F^+ by using the following additional rules.
  - If $\alpha \rightarrow b$ holds and $\alpha \rightarrow \gamma$ holds, then $\alpha \rightarrow b\,\gamma$ holds **(union)**
  - If $\alpha \rightarrow b\,\gamma$ holds, then $\alpha \rightarrow b$ holds and $\alpha \rightarrow \gamma$ holds **(decomposition)**
  - If $\alpha \rightarrow b$ holds and $\gamma\, b \rightarrow \delta$ holds, then $\alpha\,\gamma \rightarrow \delta$ holds **(pseudotransitivity)**

  The above rules can be inferred from Armstrong's axioms.

Closure of Attribute Sets

- Given a set of attributes α, define the *closure* of α under F (denoted by α^+) as the set of attributes that are functionally determined by α under F:

$$\alpha \rightarrow \beta \text{ is in } F^+ \Leftrightarrow \beta \subseteq \alpha^+$$

- Algorithm to compute α^+, the closure of α under F

```
result := α;
while (changes to result) do
    for each β → γ in F do
      begin
        if β ⊆ result then  result := result ∪ γ
      end
```

Example of Attribute Set Closure

- $R = (A, B, C, G, H, I)$
- $F = \{A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- $(AG)^+$
  1. $result = AG$
  2. $result = ABCG$ $(A \rightarrow C$ and $A \rightarrow B)$
  3. $result = ABCGH$ $(CG \rightarrow H$ and $CG \subseteq AGBC)$
  4. $result = ABCGHI$ $(CG \rightarrow I$ and $CG \subseteq AGBCH)$
- Is AG a candidate key?
  1. Is AG a super key?
     1. Does $AG \rightarrow R$? == Is $(AG)^+ \supseteq R$
  2. Is any subset of AG a superkey?
     1. Does $A \rightarrow R$? == Is $(A)^+ \supseteq R$
     2. Does $G \rightarrow R$? == Is $(G)^+ \supseteq R$

Uses of Attribute Closure

There are several uses of the attribute closure algorithm:

- Testing for superkey:
  - To test if α is a superkey, we compute α^+, and check if α^+ contains all attributes of *R*.
- Testing functional dependencies
  - To check if a functional dependency $\alpha \rightarrow \beta$ holds (or, in other words, is in F^+), just check if $\beta \subseteq \alpha^+$.
  - That is, we compute α^+ by using attribute closure, and then check if it contains β.
  - Is a simple and cheap test, and very useful
- Computing closure of F
  - For each $\gamma \subseteq R$, we find the closure γ^+, and for each $S \subseteq \gamma^+$, we output a functional dependency $\gamma \rightarrow S$.

Canonical Cover

- Sets of functional dependencies may have redundant dependencies that can be inferred from the others
  - Eg: $A \rightarrow C$ is redundant in: $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow C\}$
  - Parts of a functional dependency may be redundant
    - E.g. on RHS: $\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow CD\}$ can be simplified to

      $$\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow D\}$$
    - E.g. on LHS: $\{A \rightarrow B, \quad B \rightarrow C, \quad AC \rightarrow D\}$ can be simplified to

      $$\{A \rightarrow B, \quad B \rightarrow C, \quad A \rightarrow D\}$$
- Intuitively, a canonical cover of F is a "minimal" set of functional dependencies equivalent to F, having no redundant dependencies or redundant parts of dependencies

Extraneous Attributes

- Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F.
  - Attribute A is extraneous in α if $A \in \alpha$ and F logically implies $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$.
  - Attribute A is extraneous in β if $A \in \beta$ and the set of functional dependencies $(F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ logically implies F.
- *Note:* implication in the opposite direction is trivial in each of the cases above, since a "stronger" functional dependency always implies a weaker one
- Example: Given $F = \{A \rightarrow C, AB \rightarrow C\}$
  - B is extraneous in $AB \rightarrow C$ because $\{A \rightarrow C, AB \rightarrow C\}$ logically implies $A \rightarrow C$ (I.e. the result of dropping B from $AB \rightarrow C$).
- Example: Given $F = \{A \rightarrow C, AB \rightarrow CD\}$
  - C is extraneous in $AB \rightarrow CD$ since $AB \rightarrow C$ can be inferred even after deleting C

Testing if an Attribute is Extraneous

- Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F.
- To test if attribute $A \in \alpha$ is extraneous in α
  1. compute $(\{\alpha\} - A)^+$ using the dependencies in F
  2. check that $(\{\alpha\} - A)^+$ contains A; if it does, A is extraneous
- To test if attribute $A \in \beta$ is extraneous in β
  1. compute α^+ using only the dependencies in
     $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$,
  2. check that α^+ contains A; if it does, A is extraneous

Canonical Cover

- A *canonical cover* for F is a set of dependencies F_c such that
  - F logically implies all dependencies in $F_{c,}$ and
  - F_c logically implies all dependencies in $F,$ and
  - No functional dependency in F_c contains an extraneous attribute, and
  - Each left side of functional dependency in F_c is unique.
- To compute a canonical cover for F:

repeat

Use the union rule to replace any dependencies in F

$\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_1$ with $\alpha_1 \rightarrow \beta_1 \beta_2$

Find a functional dependency $\alpha \rightarrow \beta$ with an extraneous attribute either in α or in β

If an extraneous attribute is found, delete it from $\alpha \rightarrow \beta$

until F does not change

Note: Union rule may become applicable after some extraneous attributes have been deleted, so it has to be re-applied

Example of Computing a Canonical Cover

- $R = (A, B, C)$
  $F = \{A \rightarrow BC$
  $B \rightarrow C$
  $A \rightarrow B$
  $AB \rightarrow C\}$
- Combine $A \rightarrow BC$ and $A \rightarrow B$ into $A \rightarrow BC$
  - Set is now $\{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$
- A is extraneous in $AB \rightarrow C$
  - Check if the result of deleting A from $AB \rightarrow C$ is implied by the other dependencies
    - Yes: in fact, $B \rightarrow C$ is already present!
  - Set is now $\{A \rightarrow BC, B \rightarrow C\}$
- C is extraneous in $A \rightarrow BC$
  - Check if $A \rightarrow C$ is logically implied by $A \rightarrow B$ and the other dependencies
    - Yes: using transitivity on $A \rightarrow B$ *and* $B \rightarrow C$.
      - Can use attribute closure of A in more complex cases
- The canonical cover is: $A \rightarrow B$
  $B \rightarrow C$