Manipulating Functional Dependencies

Zaki Malik

September 30, 2008
Definition of Functional Dependency

• If \( t \) is a tuple in a relation \( R \) and \( A \) is an attribute of \( R \), then \( t_A \) is the value of attribute \( A \) in tuple \( t \).

• The FD AdvisorId \( \rightarrow \) AdvisorName holds in \( R \) if in every instance of \( R \), for every pair of tuples \( t \) and \( u \)

\[
\text{if } t_{\text{AdvisorId}} = u_{\text{AdvisorId}}, \text{ then } t_{\text{AdvisorName}} = u_{\text{AdvisorName}}
\]
Rules for Manipulating FDs

• Learn how to reason about FDs.
• Define rules for deriving new FDs from a given set of FDs.
• Use these rules to remove “anomalies” from relational designs.
• **Example:** A relation R with attributes A, B, and C, satisfies the FDs $A \rightarrow B$ and $B \rightarrow C$. What other FDs does it satisfy?
  
  $A \rightarrow C$

• What is the key for R?
  
  – A, because $A \rightarrow B$ and $A \rightarrow C$
Splitting and Combining FDs

• Can we split and combine left hand sides of FDs?

For the relation Courses is the FD

\[ \text{Number} \rightarrow \text{DeptName} \rightarrow \text{CourseName} \]

equivalent to the set of FDs

\[ \{ \text{Number} \rightarrow \text{CourseName}, \text{DeptName} \rightarrow \text{CourseName} \} \]?

– No!
Triviality of FDs

An FD $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$ is

- *trivial* if the $B$’s are a subset of the $A$’s,
  $\{B_1, B_2, \ldots B_n\} \subseteq \{A_1, A_2, \ldots A_n\}$

- *non-trivial* if at least one $B$ is not among the $A$’s,
  $\{B_1, B_2, \ldots B_n\} - \{A_1, A_2, \ldots A_n\} \neq \emptyset$

- *completely non-trivial* if none of the $B$’s are among the $A$’s, i.e.,
  $\{B_1, B_2, \ldots B_n\} \cap \{A_1, A_2, \ldots A_n\} = \emptyset$.

- *Trivial dependency rule:* The FD $A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m$ is equivalent to the FD $A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k$, where the $C$’s are those $B$’s that are not $A$’s, i.e.,
  $\{C_1, C_2, \ldots, C_k\} = \{B_1, B_2, \ldots, B_m\} - \{A_1, A_2, \ldots, A_n\}$.

- What good are trivial and non-trivial dependencies?
  - Trivial dependencies are always true.
  - They help simplify reasoning about FDs.
Closure of FD sets

• Given a relation schema R and set S of FDs
  – is the FD F logically implied by S?
• Example
  – R = {A,B,C,G,H,I}
  – S = A → B, A → C, CG → H, CG → I, B → H
  – would A → H be logically implied?
  – yes (you can prove this, using the definition of FD)
• Closure of S: $S^+$ = all FDs logically implied by S
• How to compute $S^+$?
  – we can use Armstrong's axioms
Armstrong's Axioms

• **Reflexivity** rule
  - \( A_1 A_2 \ldots A_n \rightarrow \text{a subset of } A_1 A_2 \ldots A_n \)

• **Augmentation** rule
  - \( A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \)
  then
  \[
  A_1 A_2 \ldots A_n \ C_1 C_2 \ldots C_k \rightarrow B_1 B_2 \ldots B_m C_1 C_2 \ldots C_k
  \]

• **Transitivity** rule
  - \( A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \) and
  \( B_1 B_2 \ldots B_m \rightarrow C_1 C_2 \ldots C_k \)
  then
  \[
  A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_k
  \]
Inferring $S^+$ using Armstrong's Axioms

- $S^+ = S$
- Loop
  - For each $F$ in $S$, apply reflexivity and augmentation rules
  - add the new FDs to $S^+$
  - For each pair of FDs in $S$, apply the transitivity rule
  - add the new FD to $S^+$
- Until $S^+$ does not change any further
Additional Rules

• **Union** rule
  - \( X \rightarrow Y \) and \( X \rightarrow Z \), then \( X \rightarrow YZ \)
  - (\( X, Y, Z \) are sets of attributes)

• **Decomposition** rule
  - \( X \rightarrow YZ \), then \( X \rightarrow Y \) and \( X \rightarrow Z \)

• **Pseudo-transitivity** rule
  - \( X \rightarrow Y \) and \( YZ \rightarrow U \), then \( XZ \rightarrow U \)

• These rules can be inferred from Armstrong's axioms
Example

  
  $F = \{ A \rightarrow B, A \rightarrow C, CG \rightarrow H, CG \rightarrow I, B \rightarrow H \}$

- 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
Closures of Attributes

Suppose a relation with attributes $A$, $B$, $C$, $D$, $E$, and $F$ satisfies the FDs

$$AB \rightarrow C \quad BC \rightarrow AD \quad D \rightarrow E, \quad CF \rightarrow B$$

Given these FDs,

- what is the set $X$ of attributes such that $AB \rightarrow X$ is true?
  $X = \{A, B, C, D, E\}$, i.e., $AB \rightarrow ABCDE$.

- what is the set $Y$ of attributes such that $BCF \rightarrow Y$ is true?
  $Y = \{A, B, C, D, E, F\}$, i.e., $BCF \rightarrow ABCDEF$.

- $\{B, C, F\}$ is a superkey.
Closures of Attributes: Definition

Given

- a set of attributes \( \{A_1, A_2, \ldots, A_n\} \) and
- a set of FDs \( S \),

the closure of \( \{A_1, A_2, \ldots, A_n\} \) under the FDs in \( S \) is

- the set of attributes \( \{B_1, B_2, \ldots, B_m\} \) such that for \( 1 \leq i \leq m \), the FD \( A_1 A_2 \ldots A_n \rightarrow B_i \) follows from \( S \).
- the closure is denoted by \( \{A_1, A_2, \ldots, A_n\}^+ \).

- Which attributes must \( \{A_1, A_2, \ldots, A_n\}^+ \) contain at a minimum? \( \{A_1, A_2, \ldots, A_n\} \). Why?

\[ A_1 A_2 \ldots A_n \rightarrow A_i \] is a trivial FD.
Closures of Attributes: Algorithm

Given

▸ a set of attributes \( \{A_1, A_2, \ldots, A_n\} \) and
▸ a set of FDs \( S \),
▸ compute \( X = \{A_1, A_2, \ldots, A_n\}^+ \).

1. Set \( X \leftarrow \{A_1, A_2, \ldots, A_n\} \).
2. Find an FD \( B_1 B_2 \ldots B_k \rightarrow C \) in \( S \) such that \( \{B_1, B_2, \ldots B_k\} \subseteq X \) but \( C \not\in X \).
3. Add \( C \) to \( X \).
4. Repeat the last two steps until you cannot find such an attribute \( C \).
5. The final value of \( X \) is the desired closure.
Closures of Attributes: Algorithm

- **Basis:** $Y^+ = Y$
- **Induction:** Look for an FD’s left side $X$ that is a subset of the current $Y^+$
  - If the FD is $X \rightarrow A$, add $A$ to $Y^+$
Diagrammatically:
Why is the Concept of Closures Useful?

- Closures allow us to prove correctness of rules for manipulating FDs.
  - Transitive rule: if
    \[ A_1 A_2 \ldots A_n \rightarrow B_1 B_2 \ldots B_m \]
    and
    \[ B_1 B_2 \ldots B_m \rightarrow C_1 C_2 \ldots C_n \]
    then
    \[ A_1 A_2 \ldots A_n \rightarrow C_1 C_2 \ldots C_n. \]
  - To prove this rule, simply check if
    \[ \{C_1, C_2, \ldots, C_n\} \subseteq \{A_1, A_2, \ldots, A_n\}^+. \]
- Closures allow us to procedurally define keys. A set of attributes \( X \) is a key for a relation \( R \) if and only if
  - \( \{X\}^+ \) is the set of all attributes of \( R \) and
  - for no attribute \( A \in X \) is \( \{X - \{A\}\}^+ \) the set of all attributes of \( R \).
Uses of Attribute Closure

There are several uses of the attribute closure algorithm:

• **Testing for superkey:**
  – To test if $\alpha$ is a superkey, we compute $\alpha^+$, and check if $\alpha^+$ 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 $\alpha^+$ by using attribute closure, and then check if it contains $\beta$.
  – Is a simple and cheap test, and very useful

• **Computing closure of $F$**
  – For each $\gamma \subseteq R$, we find the closure $\gamma^+$, and for each $S \subseteq \gamma^+$, we output a functional dependency $\gamma \rightarrow S$. 
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. \( \text{result} = AG \)
  2. \((A \rightarrow C \text{ and } A \rightarrow B) \ \text{result} = ABCG \)
  3. \((CG \rightarrow H \text{ and } CG \subseteq AGBC) \ \text{result} = ABCGH \)
  4. \((CG \rightarrow I \text{ and } CG \subseteq AGBCH) \ \text{result} = ABCGHI \)

- Is AG a super key?
- Is AG a key?
  1. Does \( A^+ \rightarrow R \)?
  2. Does \( G^+ \rightarrow R \)?
Example of Closure Computation

- Consider the “bad” relation Students(Id, Name, AdvisorId, AdvisorName, FavouriteAdvisorId).
- What are the FDs that hold in this relation?
  
  - Id → Name
  - Id → FavouriteAdvisorId
  - AdvisorId → AdvisorName

- To compute the key for this relation,
  
  1. Compute the closures for all sets of attributes.
  2. Find the minimal set of attributes whose closure is the set of all attributes.