Functional-Dependency Theory Roadmap

- We now consider the formal theory that tells us which functional dependencies are implied logically by a given set of functional dependencies.
- We then develop algorithms to generate lossless decompositions into BCNF and 3NF
- We then develop algorithms to test if a decomposition is dependency-preserving

FD Theory

- Closure of Set of Functional Dependencies $\mathbf{F^+}$
- Closure of α set of Attributes **α+**
- Cover of F set of functional dependencies $\mathbf{F_c}$

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.
  - 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 $\boldsymbol{F^+}$.

Closure of a Set of Functional Dependencies

- We can compute F^+, the closure of F, by repeatedly applying **Armstrong's Axioms:**
  - **Reflexive rule:** if $\beta \subseteq \alpha$, then $\alpha \rightarrow \beta$
  - **Augmentation rule:** if $\alpha \rightarrow \beta$, then $\gamma\,\alpha \rightarrow \gamma\,\beta$
  - **Transitivity rule:** if $\alpha \rightarrow \beta$, and $\beta \rightarrow \gamma$, then $\alpha \rightarrow \gamma$
- These rules are
  - **Sound** -- generate only functional dependencies that actually hold, and
  - **Complete** -- generate all functional dependencies that hold.

Example of F^+

- $R = (A, B, C, G, H, I)$
  $F = \{ A \rightarrow B$
  $A \rightarrow C$
  $CG \rightarrow H$
  $CG \rightarrow I$
  $B \rightarrow H\}$
- 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 $CG \rightarrow I$
  - $CG \rightarrow HI$
    - by augmenting $CG \rightarrow I$ to infer $CG \rightarrow CGI$,
      and augmenting of $CG \rightarrow H$ to infer $CGI \rightarrow HI$,
      and then transitivity

Closure of Functional Dependencies (Cont.)

- Additional rules:
  - **Union rule**: If $\alpha \to \beta$ holds *a*nd $\alpha \to \gamma$ holds, then $\alpha \to \beta\gamma$ holds.
  - **Decomposition rule**: If $\alpha \to \beta\gamma$ holds, then $\alpha \to \beta$ holds and $\alpha \to \gamma$ holds.
  - **Pseudotransitivity rule**:If $\alpha \to \beta$ holds *a*nd $\gamma\beta \to \delta$ holds, then $\alpha\gamma \to \delta$ holds**.**
- The above rules can be inferred from Armstrong's axioms**.**

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 shall see an alternative procedure for this task later

Closure of set of Functional Dependencies F^+

F ={A->C,AC->D,E->AD,E->H}
G= {A_>CD, E-> AH}

Evaluate: **F implies G and/or vice versa**
Hint: Given F prove G and vice versa

Closure of set of Functional Dependencies F^+

F ={A->C,AC->D,E->AD,E->H}
G= {A_>CD, E-> AH}

Evaluate: **F implies G and/or vice versa**
Hint: Given F prove G and vice versa

A->C, AC->D implies A->D
A->D & A->C implies **A->CD**

E->AD implies E->A, E->D
{E->H} U {E->A} = **{E->AH}**

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

result := α;
while (changes to *result*) **do**
  for each $\beta \rightarrow \gamma$ **in** *F* **do**
    begin
      if $\beta \subseteq$ *result* **then** *result* := *result* $\cup\ \gamma$
    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 R $\supseteq (AG)^+$
  2. Is any subset of AG a superkey?
     1. Does $A \rightarrow R$? == Is R $\supseteq (A)^+$
     2. Does $G \rightarrow R$? == Is R $\supseteq (G)^+$
     3. In general: check for each subset of size n-1

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 $\alpha^{+,}$ 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 $\gamma \rightarrow S$.

Attribute Set Closure

R (A,B,C,D,E)
F= {A->BC, CD->E, B->D, E->A}

Find Candidate Key CK of R
Hint: Find Attribute closure for each subset of R.

Attribute Set Closure

R (A,B,C,D,E)
F= {A->BC, CD->E, B->D, E->A}

Find Candidate Key CK of R
Hint: Find Attribute closure for each subset of R.

A->BC implies A->B, A->B and B->D implies A->D
A->BC implies A->C, A->C and A->D implies A->CD
A-CD and CD->E implies A->E **A->ABCDE**

E->A implies **E-> ABCDE**

CD->E implies **CD->ABCDE**
No subset of CD implies ABCDE

B->D , BC->CD implies **BC->ABCDE**
No subset of BC implies ABCDE

CK: A, E, CD, BC

Attribute Set Closure

R(A,B,C,D,E)
F= {A->C, B->E,DE->C}
R1(A,B,C)

What FDs hold on R1?
Hint: Find attribute closure of each subset of R1

Attribute Set Closure

R(A,B,C,D,E)
F= {A->D, B->E,DE->C}
R1(A,B,C)

What FDs hold on R1?
Hint: Find attribute closure of each subset of R1

Calculate attribute closures of:
A,B,C
AB,BC,AC
ABC

{AB}+ adds AB-> C to FDs on R1.

Ans. AB->C

Canonical Cover

- Suppose that we have a set of functional dependencies *F* on a relation schema. Whenever a user performs an update on the relation, the database system must ensure that the update does not violate any functional dependencies; that is, all the functional dependencies in *F* are satisfied in the new database state.
- If an update violates any functional dependencies in the set *F,* the system must roll back the update.
- We can reduce the effort spent in checking for violations by testing a simplified set of functional dependencies that has the same closure as the given set.
- This simplified set is termed the **canonical cover**
- To define canonical cover we must first define **extraneous attributes.**
  - An attribute of a functional dependency in *F* is **extraneous** if we can remove it without changing F^+

Extraneous Attributes

- Removing an attribute from the left side of a functional dependency could make it a stronger constraint.
  - For example, if we have $AB \rightarrow C$ and remove B, we get the possibly stronger result $A \rightarrow C$. It may be stronger because $A \rightarrow C$ logically implies $AB \rightarrow C$, but $AB \rightarrow C$ does not, on its own, logically imply $A \rightarrow C$
- But, depending on what our set F of functional dependencies happens to be, we may be able to remove B from $AB \rightarrow C$ safely.
  - For example, suppose that
  - $F = \{AB \rightarrow C, A \rightarrow D, D \rightarrow C\}$
  - Then we can show that F logically implies $A \rightarrow C$, making B extraneous in $AB \rightarrow C$.

Extraneous Attributes (Cont.)

- Removing an attribute from the right side of a functional dependency could make it a weaker constraint.
  - For example, if we have $AB \rightarrow CD$ and remove C, we get the possibly weaker result $AB \rightarrow D$. It may be weaker because using just $AB \rightarrow D$, we can no longer infer $AB \rightarrow C$.
- But, depending on what our set F of functional dependencies happens to be, we may be able to remove C from $AB \rightarrow CD$ safely.
  - For example, suppose that

    $F = \{ AB \rightarrow CD, A \rightarrow C$.
  - Then we can show that even after replacing $AB \rightarrow CD$ by $AB \rightarrow D$, we can still infer $AB \rightarrow C$ and thus $AB \rightarrow CD$.

Extraneous Attributes

- An attribute of a functional dependency in F is **extraneous** if we can remove it without changing F^+
- Consider a set F of functional dependencies and the functional dependency $\alpha \rightarrow \beta$ in F.
  - **Remove from the left side**: Attribute A is **extraneous** in α if
    - $A \in \alpha$ and
    - F logically implies $(F - \{\alpha \rightarrow \beta\}) \cup \{(\alpha - A) \rightarrow \beta\}$.
  - **Remove from the right side**: Attribute A is **extraneous** in β if
    - $A \in \beta$ and
    - The set of functional dependencies

      $F' = (F - \{\alpha \rightarrow \beta\}) \cup \{\alpha \rightarrow (\beta - A)\}$ 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

Testing if an Attribute is Extraneous

- Let *R* be a relation schema and let *F* be a set of functional dependencies that hold on *R* . Consider an attribute in the functional dependency $\alpha \to \beta$.
- To test if attribute $A \in \beta$ is extraneous in β
  - Consider the set:

    $$F' = (F - \{\alpha \to \beta\}) \cup \{\alpha \to (\beta - A)\},$$

  - check that α^+ contains *A;* if it does, *A* is extraneous in β
  - **$(\alpha^+)_{F'}$**

- To test if attribute $A \in \alpha$ is extraneous in α
  - Let $\gamma = \alpha - \{A\}$. Check if $\gamma \to \beta$ can be inferred from *F.*
    - Compute γ^+ using the dependencies in *F*
    - If γ^+ includes all attributes in β then , *A* is extraneous in α
    - **$(\gamma^+)_F$**

Examples of Extraneous Attributes

- Let $F = \{AB \rightarrow CD, A \rightarrow E, E \rightarrow C\}$
- To check if C is extraneous in $AB \rightarrow CD$, we:
  - Compute the attribute closure of AB under $F' = \{AB \rightarrow D, A \rightarrow E, E \rightarrow C\}$
  - The closure is $ABCDE$, which includes CD
  - This implies that C 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. That is, there are no two dependencies in F_c

  $\alpha_1 \rightarrow \beta_1$ and $\alpha_2 \rightarrow \beta_2$ such that $\alpha_1 = \alpha_2$

Canonical Cover

- To compute a canonical cover for *F*:

  $F_c = F$

  repeat

  Use the union rule to replace any dependencies in F_c of the form

  $$\alpha_1 \rightarrow \beta_1 \text{ and } \alpha_1 \rightarrow \beta_2 \text{ with } \alpha_1 \rightarrow \beta_1 \beta_2$$

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

  /* Note: test for extraneous attributes done using F_c, not F*/

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

  until (F_c does not change)

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

Example: 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: $A \rightarrow B$
  $B \rightarrow C$

Example: Computing a Canonical Cover

- **$F = \{A \rightarrow BC, B \rightarrow AC, C \rightarrow AB\}$**
- Check A->BC
- Both B and C are extraneous under F
- Algorithm picks up one of these two

- If C is deleted, we get the set $F' = \{A \rightarrow B, B \rightarrow AC$, and $C \rightarrow AB\}$. Now, B is not extraneous on the right side of $A \rightarrow B$ under F'. Continuing the algorithm, we find A and B are extraneous in the right side of $C \rightarrow AB$, leading to two choices

  of canonical cover:
- $F_c = \{A \rightarrow B, B \rightarrow C, C \rightarrow A\}$
- $F_c = \{A \rightarrow B, B \rightarrow AC, C \rightarrow B\}$.

- If B is deleted, we get the set $\{A \rightarrow C, B \rightarrow AC$, and $C \rightarrow AB\}$. This case

  is symmetrical to the previous case, leading to two more choices of canonical

  cover:
- $F_c = \{A \rightarrow C, C \rightarrow B$, and $B \rightarrow A\}$
- $F_c = \{A \rightarrow C, B \rightarrow C$, and $C \rightarrow AB\}$.

Dependency Preservation

- Let F_i be the set of dependencies F^+ that include only attributes in R_i.
  - A decomposition is **dependency preserving**, if

$$(F_1 \cup F_2 \cup \ldots \cup F_n)^+ = F^+$$

$$(F')^+ = F^+$$

- Since all functional dependencies in a restriction involve attributes of only one relation schema, it is possible to test such a dependency for satisfaction by checking only one relation.
- Using the above definition, testing for dependency preservation take exponential time.
- Not that if a decomposition is NOT dependency preserving then checking updates for violation of functional dependencies may require computing joins, which is expensive.

Dependency Preservation (Cont.)

- Let F be the set of dependencies on schema R and let R_1, R_2 , .., R_n be a decomposition of R.
- The restriction of F to R_i is the set F_i of all functional dependencies in F^+ that include **only** attributes of R_i .
- Since all functional dependencies in a restriction involve attributes of only one relation schema, it is possible to test such a dependency for satisfaction by checking only one relation.
- Note that the definition of restriction uses all dependencies in in F^+, not just those in F.
- The set of restrictions F_1, F_2 , .. , F_n is the set of functional dependencies that can be checked efficiently.

Testing for Dependency Preservation

- To check if a dependency $\alpha \rightarrow \beta$ is preserved in a decomposition of R into R_1, R_2, …, R_n , we apply the following test (with attribute closure done with respect to F)
  - *result* = α
    repeat
      for each R_i in the decomposition
        $t = (result \cap R_i)^+ \cap R_i$
        $result = result \cup t$
     until (*result* does not change)
  - If *result* contains all attributes in β, then the functional dependency $\alpha \rightarrow \beta$ is preserved.
- We apply the test on all dependencies in F to check if a decomposition is dependency preserving
- This procedure takes polynomial time, instead of the exponential time required to compute F^+ and $(F_1 \cup F_2 \cup \ldots \cup F_n)^+$

Example

- $R = (A, B, C)$
  $F = \{A \rightarrow B$
  $\quad B \rightarrow C\}$
  Key $= \{A\}$
- R is not in BCNF
- Decomposition $R_1 = (A, B)$, $R_2 = (B, C)$
  - R_1 and R_2 in BCNF
  - Lossless-join decomposition
  - Dependency preserving