Chapter 8: Relational Database Design

Database System Concepts, 6th Ed.

Edited by Radhika Sukapuram

Equivalence of a Pair of Sets of FDs

- F: A->C, AC->D, E->AD, E->H
- G: A->CD, E->AH

$A^+ = ACD$

Equivalence of a Pair of Sets of FDs

- P: A->B, AB->C, D-> ACE
- Q-> A->BC, D->AE

Canonical Cover

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

Why find the canonical cover ?

- When a relation is updated, the DBMS must check that there are no FD violations
- Checking with the canonical cover of F is efficient

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:

  F_c= F

  repeat

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

  $\alpha_1 \to \beta_1$ and $\alpha_1 \to \beta_2$ with $\alpha_1 \to \beta_1 \beta_2$

  Find a functional dependency $\alpha \to \beta$ 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 $\alpha \to \beta$ 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

Computing a Canonical Cover

- $R = (A, B, C)$
  $F = \{A \rightarrow BC$
  $B \rightarrow C$
  $A \rightarrow B$
  $AB \rightarrow C\}$

- $F_c = F$
  repeat
  Use the union rule to replace any dependencies in F_c of the form
  $\alpha_1 \rightarrow \beta_1$ and $\alpha_1 \rightarrow \beta_2$ 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)

Computing a Canonical Cover

- $R = (A, B, C)$
  $F = \{A \rightarrow BC$
  $B \rightarrow C$
  $A \rightarrow B$
  $AB \rightarrow C\}$

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 $\{A \rightarrow BC, B \rightarrow C, AB \rightarrow C\}$

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 $AB \rightarrow C$
  - *as F logically implies (F-{AB->C}) ∪ {B->C} as B → C already exists*
  - Set is now $\{A \rightarrow BC, B \rightarrow C\}$

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 $AB \rightarrow C$
  - *as F logically implies (F-{AB->C}) ∪ {B->C} as* $B \rightarrow C$ *already exists*
  - Set is now $\{A \rightarrow BC, B \rightarrow C\}$
- C is extraneous in $A \rightarrow BC$ as
  - $A \rightarrow BC$ *is* $A \rightarrow B$, $A \rightarrow C$.
  - *Now* $A \rightarrow B$ *and* $B \rightarrow C$ *are sufficient to imply* $A \rightarrow C$
  - *Therefore C is dropped from* $A \rightarrow BC$
    - Can use attribute closure of A in more complex cases
- The canonical cover is: $A \rightarrow B$
  $B \rightarrow C$

Lossless-join Decomposition

- For the case of $R = (R_1, R_2)$, we require that for all possible relations r on schema R

$$r = \prod_{R1}(r) \bowtie \prod_{R2}(r)$$

- A decomposition of R into R_1 and R_2 is lossless join if at least one of the following dependencies is in F^+:
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

  In other words, $R_1 \cap R_2$ forms a superkey of R1 or R2

- The above functional dependencies are a sufficient condition for lossless join decomposition
  - the dependencies are a necessary condition only if all constraints are functional dependencies (there are constraints other than FDs)

Example

- $R = (A, B, C)$
  $F = \{A \rightarrow B, B \rightarrow C)$
  - Can be decomposed in two different ways
- $R_1 = (A, B), \quad R_2 = (B, C)$
  - Lossless-join decomposition:

    $$R_1 \cap R_2 = \{B\} \text{ and } B \rightarrow BC$$

  - Dependency preserving

$\bowtie$

Example

- $R = (A, B, C)$
  $F = \{A \rightarrow B, B \rightarrow C)$
  - Can be decomposed in two different ways
- $R_1 = (A, B), \quad R_2 = (B, C)$
  - Lossless-join decomposition:

    $$R_1 \cap R_2 = \{B\} \text{ and } B \rightarrow BC$$

  - Dependency preserving
- $R_1 = (A, B), \quad R_2 = (A, C)$
  - Lossless-join decomposition:

    $$R_1 \cap R_2 = \{A\} \text{ and } A \rightarrow AB$$

  - Not dependency preserving
    (cannot check $B \rightarrow C$ without computing $R_1 \bowtie R_2$)

Example cont.

- *instr_dept* (*ID, name, salary, dept_name, building, budget*)
  - *instructor* (*ID, name, salary, dept_name)*
  - *department(dept_name, building, budget*)

Is this decomposition lossless ?

Testing for Dependency Preservation

- To check if a dependency $\alpha \rightarrow \beta$ is preserved in a decomposition of R into $R_1, R_2, \ldots, R_n$ 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