Chapter 8: Relational Database Design

Database System Concepts, 6th Ed.

Edited by Radhika Sukapuram

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,D,G)
- F = {A->B, B->C, A->D, D->G}

- Case: R1(A,B), R2(B,C,D), R3(D,G), Whether A->G?

Testing for BCNF

- To check if a non-trivial dependency $\alpha \rightarrow \beta$ causes a violation of BCNF
  1. compute α^+ (the attribute closure of α), and
  2. verify that it includes all attributes of *R*, that is, it is a superkey of *R*.

- **Simplified test**: To check if a relation schema *R* is in BCNF,
  - it suffices to check only the dependencies in the given set *F* for violation of BCNF, rather than checking all dependencies in F^+.
  - If none of the dependencies in *F* causes a violation of BCNF, then none of the dependencies in F^+ will cause a violation of BCNF either.

Testing for BCNF

- However, **simplified test using only *F* is incorrect when testing a relation in a decomposition of R**
  - Consider $R = (A, B, C, D, E)$, with $F = \{ A \rightarrow B, BC \rightarrow D\}$
    - Decompose R into $R_1 = (A,B)$ and $R_2 = (A,C,D, E)$
    - Neither of the dependencies in F contain only attributes from (A,C,D,E) so we might be mislead into thinking R_2 satisfies BCNF.
    - In fact, dependency $AC \rightarrow D$ in F^+ shows R_2 is not in BCNF.
    - So we need to check with all dependencies in F^+ to check if a decomposition is in BCNF.

Easier :Testing Decomposition for BCNF

- To check if a relation R_i in a decomposition of R is in BCNF,
  - Either
    - Find F^+
    - Find **restriction** of F to R_i (that is, all FDs in F^+ that contain only attributes from R_i)
    - test R_i for BCNF with respect to the restriction of F to R_i
  - or **use the original set of dependencies *F* that holds on *R***, but with the following test:
    - for every set of attributes $\alpha \subseteq R_i$, check that α^+ (the attribute closure of α under F)
      - » either includes no attribute of $R_i - \alpha$,
      - » or includes all attributes of R_i.
    - If the condition is violated by some α in R_i, the dependency
      $\alpha \rightarrow (\alpha^+ - \alpha) \cap R_i$
      can be shown to hold on R_i, and R_i violates BCNF.

BCNF Decomposition Algorithm

result := {R };
done := false;
compute F^+;
while (not *done)* **do**
  if (there is a schema R_i in *result* that is not in BCNF)
    then begin
      let $\alpha \rightarrow \beta$ be a nontrivial functional dependency that
        holds on R_i such that $\alpha \rightarrow R_i$ is not in F^+,
        and $\alpha \cap \beta = \emptyset$;
      result := $(result - R_i) \cup (R_i - \beta) \cup (\alpha, \beta)$;
    end
  else *done* := **true;**

Note: each R_i is in BCNF, and decomposition is lossless-join.

Example of BCNF Decomposition

- $R = (A, B, C)$
  $F = \{B \rightarrow C, A \rightarrow B\}$
  Key = $\{A\}$
- R is not in BCNF ($B \rightarrow C$ but B is not a superkey)
- $F+ = \{B \rightarrow C, A \rightarrow B$
  $A \rightarrow C\}$
- $B \rightarrow C$ *is an FD such that* $B \rightarrow$ ABC is not in F+ and $B \cap C = \varnothing$
- So remove ABC, add AC and BC
- Decomposition
  - $R_1 = (B, C)$
  - $R_2 = (A,B)$

Example of BCNF Decomposition

- *class* (*course_id*, *title*, *dept_name*, *credits*, *sec_id*, *semester*, *year*, *building*, *room_number*, *capacity*, *time_slot_id*)
- Functional dependencies:
  - *course_id*→ *title*, *dept_name*, *credits*
  - *building*, *room_number*→*capacity*
  - *course_id*, *sec_id*, *semester*, *year*→*building*, *room_number*, *time_slot_id*
- A candidate key {*course_id*, *sec_id*, *semester*, *year*}.
- BCNF Decomposition:
  - *course_id*→ *title*, *dept_name*, *credits* holds
    - but *course_id* is not a superkey. *class* is not in BCNF
  - We replace *class* by:
    - *course*(*course_id*, *title*, *dept_name*, *credits*)
    - *class-1* (*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number, capacity*, *time_slot_id*)

BCNF Decomposition (Cont.)

- *course* is in BCNF
  - How do we know this ?
- *building*, *room_number*→*capacity* holds on *class-1*
  - but {*building*, *room_number*} is not a superkey for *class-1*.
  - We replace *class-1* by:
    - *classroom* (*building*, *room_number*, *capacity*)
    - *section* (*course_id*, *sec_id*, *semester*, *year*, *building*, *room_number*, *time_slot_id*)
- *classroom* and *section* are in BCNF.

BCNF and Dependency Preservation

It is not always possible to get a BCNF decomposition that is dependency preserving

- $R = (J, K, L)$
  $F = \{JK \rightarrow L$
  $\quad L \rightarrow K\}$
  Two candidate keys = JK and JL
- R is not in BCNF
- Any decomposition of R will fail to preserve

$$JK \rightarrow L$$

This implies that testing for $JK \rightarrow L$ requires a join

Third Normal Form: Motivation

- There are some situations where
  - BCNF is not dependency preserving, and
  - efficient checking for FD violation on updates is important
- Solution: define a weaker normal form, called Third Normal Form (3NF)
  - Allows some redundancy (with resultant problems; we will see examples later)
  - But functional dependencies can be checked on individual relations without computing a join.
  - There is always a lossless-join, dependency-preserving decomposition into 3NF.

Third Normal Form

- A relation schema R is in **third normal form (3NF)** if for all:

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

  at least one of the following holds:
  - $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
  - α is a superkey for R
  - Each attribute A in $\beta - \alpha$ is contained in a candidate key for R.

    (**NOTE**: each attribute may be in a different candidate key)
- If a relation is in BCNF it is in 3NF
  - since in BCNF one of the first two conditions above must hold.
- The third condition is a minimal relaxation of BCNF to ensure dependency preservation (will see why later).
- A relation in 3NF may have redundancies due to FDs

3NF Example

- Relation *dept_advisor*:
  - *dept_advisor* (*s_ID, i_ID, dept_name)*
    $F = \{s\_ID, dept\_name \rightarrow i\_ID,\ \ i\_ID \rightarrow dept\_name\}$
  - Two candidate keys: *s_ID, dept_name,* and *i_ID, s_ID*
  - *R* is in 3NF
    - $s\_ID, dept\_name \rightarrow i\_ID$
      - *s_ID, dept_name* is a superkey
    - $i\_ID \rightarrow dept\_name$
      - *dept_name* is contained in a candidate key

Redundancy in 3NF

- There is some redundancy in this schema
- Example of problems due to redundancy in 3NF
  - $R = (J, K, L)$
    $F = \{JK \rightarrow L, L \rightarrow K\}$

J	L	K
j_1	l_1	k_1
j_2	l_1	k_1
j_3	l_1	k_1
null	l_2	k_2

- repetition of information (e.g., the relationship l_1, k_1)
  - *(i_ID, dept_name)*
- need to use null values (e.g., to represent the tuple l_2, k_2 where there is no corresponding value for J)
  - Or do not represent this tuple at all

Testing for 3NF

- Optimization: Need to check only FDs in *F*, need not check all FDs in F^+.
- Use attribute closure to check (for each dependency $\alpha \rightarrow \beta$) whether α is a superkey.
- If α is not a superkey, we have to verify if each attribute in β is contained in a candidate key of *R*
  - this test is rather more expensive, since it involves finding candidate keys
  - testing for 3NF has been shown to be NP-hard
  - Interestingly, **decomposition** into third normal form (described shortly) can be done in polynomial time

3NF Decomposition Algorithm

Let F_c be a canonical cover for F;

i := 0;

for each functional dependency $\alpha \rightarrow \beta$ in F_c **do**

if none of the schemas R_j, $1 \leq j \leq i$ contains $\alpha\ \beta$

then begin

$i := i + 1$;

$R_i := \alpha\ \beta$

end

if none of the schemas R_j, $1 \leq j \leq i$ contains a candidate key for R

then begin

$i := i + 1$;

R_i := any candidate key for R;

end

/* Optionally, remove redundant relations */

repeat

if any schema R_j is contained in another schema R_k

then /* delete R_j ***/**

$R_j = R_i$;

$i = i - 1$;

return $(R_1, R_2, \ldots, R_i)$

3NF Decomposition Algorithm (Cont.)

- Above algorithm ensures:
  - each relation schema R_i is in 3NF
  - decomposition is dependency preserving and lossless-join

3NF Decomposition: An Example

- Relation schema:

  cust_banker_branch = (*<u>customer_id, employee_id</u>*, *branch_name*, *type*)

- The functional dependencies for this relation schema are:

  1. *customer_id*, *employee_id* $\rightarrow$ *branch_name*, *type*
  2. *employee_id* $\rightarrow$ *branch_name*
  3. *customer_id*, *branch_name* $\rightarrow$ *employee_id*

- We first compute a canonical cover

  - *branch_name* is extraneous in the r.h.s. of the 1st dependency

    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 in β

  - No other attribute is extraneous, so we get F_C =

    customer_id, *employee_id* $\rightarrow$ *type*
    employee_id $\rightarrow$ *branch_name*
    customer_id, *branch_name* $\rightarrow$ *employee_id*

3NF Decomposition Example (Cont.)

- The **for** loop generates following 3NF schema:

  (*customer_id, employee_id, type*)

  (*employee_id, branch_name*)

  (*customer_id, branch_name, employee_id)*

  - Observe that (*customer_id, employee_id, type*) contains a candidate key of the original schema, so no further relation schema needs be added

- At end of for loop, detect and delete schemas, such as (*employee_id, branch_name*), which are subsets of other schemas
  - result will not depend on the order in which FDs are considered

- The resultant 3NF schema is:

  (*customer_id, employee_id, type*)

  (*customer_id, branch_name, employee_id)*

Comparison of BCNF and 3NF

- It is always possible to decompose a relation into a set of relations that are in 3NF such that:
  - the decomposition is lossless
  - the dependencies are preserved
- It is always possible to decompose a relation into a set of relations that are in BCNF :
  - such that the decomposition is lossless
  - the dependencies may not be preserved