3NF

Relational Database Design

_Part 4b

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
- But functional dependencies can be checked on individual relations without computing a join.
- There is always a lossless-join, dependency-preserving decomposition into 3NF.

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

There is some redundancy in this schema

Redundancy in 3NF

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 relationship l_2, k_2 where there is no corresponding value for J).

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 $\alpha \rightarrow \beta$, if 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 involve finding candidate keys
- testing for 3NF has been shown to be NP-hard
- Interestingly, decomposition into third normal form 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, $1 \le j \le i$ contains $\alpha \beta$
    then begin
      $i := i + 1$;
      $R_i := \alpha \beta$
    end
if none of the schemas R_j, $1 \le j \le i$ 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 ***/**
    $R_j = R;;$
    $i=i-1;$
return $(R_1, R_2, ..., R_i)$

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 = (*customer_id, employee_id*, *branch_name, type*)

The functional dependencies for this relation schema are:

1. *customer_id, employee_id* → *branch_name, type*
2. *employee_id* → *branch_name*
3. *customer_id, branch_name* → *employee_id*

We first compute a canonical cover

- *branch_name* is extraneous in the r.h.s. of the 1st dependency
- No other attribute is extraneous, so we get F_C =

  customer_id, employee_id → *type*
  employee_id → *branch_name*
  customer_id, branch_name → *employee_id*

3NF Decompsition 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 simplified 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
- it may not be possible to preserve dependencies.

Design Goals

Goal for a relational database design is:

- BCNF.
- Lossless join.
- Dependency preservation.

If we cannot achieve this, we accept one of

- Lack of dependency preservation
- Redundancy due to use of 3NF