Relational Database Design

Instructor : Nitesh Kumar Jha

niteshjha@soa.ac.in

ITER,S'O'A(DEEMED TO BE UNIVERSITY)

August 2018

2NF(2nd Normal Form)

- A Relation schema R with FD is said to be in 2NF w.r.t F^+

If there doesn't exist any partial functional dependency in F^+ .

Partial Functional Dependency:- Consider a relation schema that has a composite key k. The relation schema is said to exhibit partial FD of the form x 'determines' y or x->y if x is a proper subset of k(key of R) and y is a non key attribute.

Alt Def: When a part of key (key being composite) functionally determines non key attribute then the functional dependency shall be a partial functional dependency.

2NF(2^{nd} Normal Form) Example

- $R = (A, B, C, D, E)$
  $F = \{A \rightarrow B, C \rightarrow D, AC \rightarrow E\}$
  Key = $\{AC\}$

Verify R satisfy 2NF or not??

Key=AC

There exist partial FDs of the form $A \rightarrow B$, $C \rightarrow D$
Hence it is not in 2NF

??? SO what should we do??

Goals of Normalization

- Let *R* be a relation scheme with a set *F* of functional dependencies.
- Decide whether a relation scheme *R* is in "good" form.
- In the case that a relation scheme *R* is not in "good" form, decompose it into a set of relation scheme $\{R_1, R_2, ..., R_n\}$ such that
  - each relation scheme is in good form
  - the decomposition is a lossless-join decomposition
  - Preferably, the decomposition should be dependency preserving.

2NF(2nd Normal Form) Example

- $R = (A, B, C, D, E)$
  $F = \{A \rightarrow B, C \rightarrow D, AC \rightarrow E\}$
  Key = $\{AC\}$

Decomposition

- $R_1 = (A, B)$ key = A $F_1=\{A \rightarrow B\}$ key = A
- $R_2 = (C, D)$ key = C $F_2=\{C \rightarrow D\}$ key = C
- $R_3 = \{A,C,E\}$ key = AC $F_3=\{AC \rightarrow E\}$ key = AC

This is lossless decomposition because

$R_1 \cap R_3 = A \rightarrow B$ *is the key of* R_1

$R_2 \cap R_3 = C \rightarrow D$ *is the key of* R_2

Its also dependency preserving because

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

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., $\beta \in \alpha$)
  - α is a super key for R
  - At least one attribute in right hand side i.e. β must contain a prime attribute (i.e. a part of candidate key)

  (**NOTE**: each attribute may be in a different candidate key

key$\rightarrow$nonkey$\rightarrow$nonkey leads to redundancy and hence must be eliminated

Alternative Definition: A relation schema(R) is in 3NF if R is in 2NF and doesn't exhibit any transitive chain of dependency of the form key$\rightarrow$nonkey$\rightarrow$nonkey

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

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 (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 ***/**
    $R_j = R$;
    $i=i-1$;
return $(R_1, R_2, ..., R_i)$

Optional---Erase Unnecessary Tables

3NF Decomposition Algorithm (Cont.)

- Summary of the Above Algorithm:

  1. To Find Canonical Cover $\boldsymbol{F_c}$ for the F.D set.
  2. For each F.D in $\boldsymbol{F_c}$ we create a new Relational schema.
  3. If any schema doesn't contain the candidate key then make a new schema for the candidate key itself.
  4. Remove unnecessary relation schemas

- Above algorithm ensures:
  - each relation schema R_i is in 3NF
  - decomposition is dependency preserving and lossless-join
  - Proof of correctness is at end of this chapter

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
  - 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*

  Key= {employee_id, customer_id}

3NF Decompsition Example (Cont.)

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

  (*customer_id, employee_id, type*)

  (<u>*employee_id*</u>, *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 (<u>*employee_id*</u>, *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)*

Redundancy in 3NF

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

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).
  - *(i_ID, dept_namel)* if there is no separate relation mapping instructors to departments

Thank You