Relational Database Design (II)

Chapter 15 in 6th Edition

10.3 Lossless and dependency-preserving decomposition into 3NF

A lossless and dependency-preserving decomposition into 3NF is always possible.

More definitions regarding FD's are needed.

A set F of FD's is minimal if

1. Every FD $X \rightarrow Y$ in F is simple: Y consists of a single attribute,
2. Every FD $X \rightarrow A$ in F is *left-reduced*: there is no proper subset $Y \subset X$ such that $X \rightarrow A$ can be replaced with $Y \rightarrow A$.
   that is, there is no $Y \subset X$ such that Iff F |= Y → A
   $$((F - \{X \rightarrow A\}) \cup \{Y \rightarrow A\})^+ = F^+$$
3. No FD in F can be removed; that is, there is no FD $X \rightarrow A$ in F such that Iff X→A is inferred From F– { X→A}
   $$(F - \{X \rightarrow A\})^+ = F^+.$$

10.3.1 Computing a minimum cover

F is a set of FD's.

A *minimal cover* (or *canonical cover*) for F is a minimal set of FD's F_{min} such that $F^+ = F^+_{min}$.

<u>Algorithm Min_Cover</u>

Input: a set F of functional dependencies.

Output: a minimum cover of F.

Step 1: *Reduce right side*. Apply Algorithm Reduce right to F.

Step 2: *Reduce left side*. Apply Algorithm Reduce left to the output of Step 2.

Step 3: *Remove redundant* FDs. Apply Algorithm Remove_redundency to the output of Step 2. The output is a minimum cover.

Below we detail the three Steps.

10.3.1 Computing a minimum cover(cont)

Algorithm Reduce_right

INPUT: F.

OUTPUT: right side reduced F'.

For each FD $X \rightarrow Y \in F$ where Y = $\{A_1, A_2, ..., A_k\}$, we use all X $\rightarrow \{A_i\}$ (for $1 \leq i \leq k$) to replace $X \rightarrow Y$.

Algorithm Reduce_left

INPUT: right side reduced F.

OUTPUT: right and left side reduced F'.

For each $X \rightarrow \{A\} \in F$ where X = $\{A_i : 1 \leq i \leq k\}$, do the following. For i = 1 to k, replace X with $X - \{A_i\}$ if A $\in (X - \{A_i\})^+$.

Algorithm Reduce_redundancy

INPUT: right and left side reduced F.

OUTPUT: a minimum cover F' of F.

For each FD $X \rightarrow \{A\} \in F$, remove it from F if: $A \in X^+$ with respect to $F - \{X \rightarrow \{A\}\}$.

Example:

R = (A, B, C, D, E, G)

F = {A $\rightarrow$ BCD, B $\rightarrow$ CDE, AC $\rightarrow$ E}

Step 1: F' = {A $\rightarrow$ B, A$\rightarrow$C, A $\rightarrow$ D, B$\rightarrow$ C, B$\rightarrow$ D, B$\rightarrow$ E, AC $\rightarrow$ E}

Step 2: AC $\rightarrow$ E

C^+ = {C}; thus C $\rightarrow$ E is not inferred by F'.

Hence, AC $\rightarrow$ E cannot be replaced by A $\rightarrow$ E.

A^+ = {A, B, C, D, E}; thus, A$\rightarrow$ E is inferred by F'.

Hence, AC$\rightarrow$ E can be replaced by A $\rightarrow$ E.

F'' = {A$\rightarrow$ B, A$\rightarrow$C, A$\rightarrow$ D, A$\rightarrow$ E, B$\rightarrow$ C, B$\rightarrow$D, B$\rightarrow$ E}

Step 3: $A+|_{F'' - \{A \rightarrow B\}}$= {A, C, D, E}; thus A$\rightarrow$B is not inferred by F'' –{A$\rightarrow$B}.

That is, A$\rightarrow$B is not redundant.

$A+|_{F'' - \{A \rightarrow C\}}$= {A, B, C, D, E}; thus, A$\rightarrow$ C is redundant.

Thus, we can remove A$\rightarrow$C from F'' to obtain F'''.

Iteratively, we can A$\rightarrow$D and A$\rightarrow$E but not the others.

Thus, F_{min}={A$\rightarrow$B, B$\rightarrow$C, B$\rightarrow$D, B$\rightarrow$E}.

10.3.2 3NF decomposition algorithm

Algorithm 3NF decomposition

1. Find a minimum cover *F'* of *F*.
2. For each left side *X* that appears in *F'*, do:

   create a relation schema $X \cup A_1 \cup A_2 ... \cup A_m$ where $X \rightarrow \{A_1\}, ... , X \rightarrow \{A_m\}$ are all the dependencies in *F'* with *X* as left side.
3. if none of the relation schemas contains a key of *R*,

   create one more relation schema that contains attributes that form a key for *R*.

See E/N Algorithm 15.4.

Example:

R = (A, B, C, D, E, G)

F_{min}={A$\rightarrow$B, B$\rightarrow$C, B$\rightarrow$D, B$\rightarrow$E}.

Candidate key: (A, G)

R_1 = (A, B), R_2 = (B, C, D, E)

R_3 = (A, G)

10.3.2 3NF decomposition algorithm(cont)

Example 6:(From Desai 6.31)

Beginning again with the *SHIPPING* relation. The functional dependencies already form a canonical cover.

- From *Ship*→*Capacity*, derive R_1(<u>*Ship*</u>,*Capacity*),
- From {*Ship*,*Date*} → *Cargo*, derive

  R_2(<u>*Ship* , *Date*</u> , *Cargo*),
- From {*Capacity*,*Cargo*} → *Value*, derive

  R_3(<u>*Capacity* , *Cargo*</u> , *Value*).
- There are no attributes not yet included and the original key {*Ship*,*Date*} is included in R_2.

10.3.2 3NF decomposition algorithm(cont)

Example 7: Apply the algorithm to the LOTS example given earlier.

A minimal cover is

{ $Property\_Id \rightarrow Lot\_No$,

$Property\_Id \rightarrow Area$, $\{City, Lot\_No\} \rightarrow Property\_Id$,

$Area \rightarrow Price$, $Area \rightarrow City$, $City \rightarrow Tax\_Rate$ }.

This gives the decomposition:

R_1 (<u>*Property_Id*</u> , *Lot_No* , *Area*)

R_2 (<u>*City* , *Lot_No*</u> , *Property_Id*)

R_3 (<u>*Area*</u> , *Price* , *City*)

R_4 (<u>*City*</u> , *Tax_Rate*)

Exercise 1: Check that this is a lossless, dependency preserving decomposition into 3NF.

Exercise 2: Develop an algorithm for computing a key of a table R with respect to a given *F* of FDs.

Summary

- Data redundancies are undesirable as they create the potential for update anomalies,
- One way to remove such redundancies is to normalise a design, guided by FD's.
- BCNF removes all redundancies due to FD's, but a dependency preserving decomposition cannot always be found,
- A dependency preserving, lossless decomposition into 3NF can always be found, but some redundancies may remain,
- Even where a dependency preserving, lossless decomposition that removes all redundancies can be found, it may not be possible, for efficiency reasons, to remove all redundancies.