3NF Decomposition

R&G Chapter 19

Features of a Good Decomposition

- **A good decomposition is**
  - Lossless
    - Can obtained by BCNF decomposition
  - Dependency preserving

Problem #2 of Decomposition: Dependency preservation

A	B	C
1	2	3
4	5	6
7	2	8

A	C
1	3
4	6
7	8

B	C
2	3
5	6
2	8

A **->** B; C **->** B

X={A, C}, Y={ B, C}, X ∩ Y = {C}, C->{B, C}

Lossless decomposition!

A	C
1	3
4	6
7	8

Join

B	C
2	3
5	6
2	8

=

A	B	C
1	2	3
4	5	6
7	2	8

But, now we can't check A **->** B without doing a join!
(Problem #2 of decomposition!)

BCNF and 3NF Decomposition

- **Some rules to remember**

	BCNF	3NF
Data redundancy	NONE	May still have some
Lossless-join decomposition	Guaranteed	Guaranteed
dependency-preserving decomposition	Not guaranteed	Guaranteed

Task #2

- **How to decompose the table so that dependency checking does not need joins?**

Dependency Preserving Decomposition

Projection of set of FDs F

If R is decomposed into X and Y, then the projection of F on X (denoted F_X) is the set of FDs U -> V in F^+ such that all of the attributes U, V are in X. *(similarly for* F_y*)*

- E.g., $F^+ = \{A\text{->}B , A\text{->}C, D\text{->}E\}$
- Two tables: X={A, B, C}, Y={C, D, E}
- What are F_X and F_y?

Dependency Preserving Decomposition

Projection of set of FDs F

If R is decomposed into X and Y, the projection of F on X (denoted F_X) is the set of FDs U -> V in F^+ such that all of the attributes U, V are in X. *(similarly for* F_y*)*

- E.g., $F^+ = \{A\text{->}B , A\text{->}C, D\text{->}E\}$
- Two tables: X={A, B, C}, Y={C, D, E}
- What are F_X and F_y?
    - $F_X = \{A\text{->}B , A\text{->}C\}$,
    - $F_y = \{D\text{->} E\}$.

Dependency Preserving Decompositions (Contd.)

- **Decomposition of R into X and Y is *dependency preserving* if $(F_X \cup F_Y)^+ = F^+$**
  - i.e., consider only dependencies in the closure F^+ that can be checked in X without considering Y, and in Y without considering X, these imply all dependencies in F^+.

Decomposition into 3NF

- The algorithm for BCNF decomposition does not ensure dependency preservation (it only assures lossless join)

- To ensure dependency preservation (and lossless-join)
  - Instead of the given set of FDs F, use a *minimal cover for F*.

Minimal Cover for a Set of FDs

- **Given a set of FDs F, G is the *minimal cover* of F if G satisfies the following three conditions:**
  1. Right hand side (RHS) of each FD in G is a single attribute;
  2. $G^+ = F^+$; and
  3. G is minimal: if we modify G by deleting an FD in G, G^+ changes.

- **Intuitively, every FD in G is needed, and ``*as small as possible*" in order to get the same closure as F.**

Finding Minimal Cover

- **Step 1: Minimize right-hand-side (RHS) of FDs so that they only contain single attributes**
  - Change X->YZ to be X->Y and X->Z
- **Step 2: Minimize left-hand-side (LHS)**
  - If there are two FDs: A->B and ABX->Z, replace ABX->Z with AX->Z
    - For example: if AB->C and A->B, then replace AB->C with A->C.
- **Step 3: Remove redundant FDs**
  - If X->Y can be inferred from other FDs, remove X->Y

Example

- **R(ABCDE)**
- **F={A->D, BC->AD, C->B, E->A, E->D}**
- **What is the minimal cover of F?**

Example - Solution

- **R(ABCDE)**
- **F={A->D, BC->AD, C->B, E->A, E->D}**
- **What is the minimal cover of F?**
- **Way of thinking:**
  - Step 1: Minimize RHS to be single attribute:
    - Split BC->AD to BC->A, BC->D
  - Step 2: Minimize LHS:
    - Since C->B, replace BC->A with C->A, and BC->D with C->D
  - Step 3: Remove redundant FDs
    - C->A, A->D => C->D, so remove C->D
    - E->A, A->D => E->D, so remove E->D
  - Minimal cover: {A->D, C->A, C->B, E->A}

3NF Decomposition

Input: a relation R with FDs F.

- Step 1: Find the minimal cover F′ of F.
- Step 2: Generate a BCNF decomposition {R1, …Rn} of R w.r.t. **F′**
- Step 3: Identify the dependencies N in F′ that is not preserved by BCNF decomposition {R1, …Rn}
  - i.e., those dependencies in F′ whose attributes are not included in any table in {R1, …Rn}
- Step 4: For each X-> Y in N, create a relation schema XY and add it to {R1…Rn}

Notes:

- Step 2 guarantees it is a lossless decomposition
- Step 3 and 4 ensure it is a dependency-preserving decomposition
  - Both steps ensure $F_{R1} \cup \ldots \cup F_{Rn} = F'$
  - Thus it must be true $(F_{R1} \cup \ldots \cup F_{Rn})^+ = F'^+$
- 3NF decomposition is guaranteed to be lossless and dependency-preserving

3NF Decomposition Example

- R(ABCDE)
- F={A->D, BC->AD, C->B, E->B, E->D}
- Decompose R in a lossless-join, dependency-preserving way (i.e., decompose R into 3NF tables!)

3NF Decomposition Example (cont.)

- **Solution:**
  - Step 1: Find minimal cover F′ of F
    - F′ = {A->D, C->A, C->B, E->B, E->D}
  - Step 2: create a lossless-join BCNF decomposition D of R
    - Step 2.1. find candidate key(s): CE
    - Step 2.2. construct BCNF decomposition based on F′: D = {BC, CE, AC, AD} (following order of A->D, C->A, C->B)
  - Step 3: Identify the dependencies N in F′ that is not preserved by D
    - N = {E->D, E->B}
  - Step 4: For each X-> Y in N, create a relation schema XY and add it to D
    - D={BC, CE, AC, AD, ED, EB}

BCNF Versus 3NF Decomposition

	BCNF	3NF
Redundancy	NONE	May still have some
Lossless-join decomposition	Guaranteed	Guaranteed
dependency-preserving decomposition	Not guaranteed	Guaranteed

- **For any given R, there is always a 3NF decomposition.**
  - But it may not have a BCNF decomposition
  - Remember, BCNF is stricter than 3NF
  - A BCNF decomposition is always a 3NF decomposition.