CZ2007 Introduction to Database Systems (Week 5)

Topic 4: Third Normal Form (2)

Dr. Ng Wee Keong
Associate Professor

This Lecture

- 3NF Decomposition
- Properties of 3NF

Exercise

- $S = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$

1. Transform the FDs to ensure that the right hand side of each FD has only one attribute
2. See if any FD can be derived from the other FDs. Remove those FDs one by one
3. Check if we can remove any attribute from the left hand side of any FD

Exercise

- $S = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$

1. Transform the FDs to ensure that the right hand side of each FD has only one attribute

- Results: $M = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$

2. See if any FD can be derived from the other FDs. Remove those FDs one by one

Exercise

- $M = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$

2. See if any FD can be derived from the other FDs. Remove those FDs one by one.

- Try $A \rightarrow C$ first
  - If $A \rightarrow C$ is removed, then the ones left would be $AC \rightarrow D$, $AD \rightarrow B$
  - With the remaining FDs, we have $\{A\}^+ = \{A\}$
  - Since $\{A\}^+$ does not contain C, we know that $A \rightarrow C$ cannot be derived from the remaining FDs
  - Therefore, $A \rightarrow C$ cannot be removed

Exercise

- M = {A→C, AC→D, AD→B}

2. See if any FD can be derived from the other FDs. Remove those FDs one by one.

- Next, try AC→D
  - If AC→D is removed, then the ones left would be A→C, AD→B
  - With the remaining FDs, we have $\{AC\}^+ = \{AC\}$
  - Since $\{AC\}^+$ does not contain D, we know that AC→D cannot be derived from the remaining FDs
  - Therefore, AC→D cannot be removed

Exercise

- $M = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$

2. See if any FD can be derived from the other FDs. Remove those FDs one by one.

- Next, try $AD \rightarrow B$
  - If $AD \rightarrow B$ is removed, then the ones left would be $A \rightarrow C$, $AC \rightarrow D$
  - With the remaining FDs, we have $\{AD\}^+ = \{AD\}$
  - Since $\{AD\}^+$ does not contain B, we know that $AD \rightarrow B$ cannot be derived from the remaining FDs
  - Therefore, $AD \rightarrow B$ cannot be removed

Exercise

- $M = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$

3. Check if we can remove any attribute from the left hand side of any FD

- First, try to remove A from $AC \rightarrow D$
  - It results in $C \rightarrow D$
  - Can $C \rightarrow D$ be derived from M?
  - $\{C\}^+ = \{C\}$ given M.
  - Since $\{C\}^+$ does not contain D, we know that $C \rightarrow D$ cannot be derived from M
  - Therefore, A cannot be removed from $AC \rightarrow D$

Exercise

- $M = \{A \rightarrow C, AC \rightarrow D, AD \rightarrow B\}$

3. Check if we can remove any attribute from the left hand side of any FD

- Next, try to remove C from $AC \rightarrow D$
  - It results in $A \rightarrow D$
  - Can $A \rightarrow D$ be derived from M?
  - $\{A\}^+ = \{ABCD\}$ given M.
  - Since $\{A\}^+$ contains D, we know that $A \rightarrow D$ can be derived from M
  - Therefore, C can be removed from $AC \rightarrow D$
- New $M = \{A \rightarrow C, A \rightarrow D, AD \rightarrow B\}$

Exercise

- New M = {A→C, A→D, AD→B}

3. Check if we can remove any attribute from the left hand side of any FD

- Next, try to remove A from AD→B
  - It results in D→B
  - Can D→B be derived from M?
  - $\{D\}^+ = \{D\}$ given M.
  - Since $\{D\}^+$ does not contain B, we know that D→B cannot be derived from M
  - Therefore, A cannot be removed from AD→B

Exercise

- $M = \{A \rightarrow C, A \rightarrow D, AD \rightarrow B\}$

3. Check if we can remove any attribute from the left hand side of any FD

- Next, try to remove D from $AD \rightarrow B$
  - It results in $A \rightarrow B$
  - Can $A \rightarrow B$ be derived from M?
  - $\{A\}^+ = \{ABCD\}$ given M.
  - Since $\{A\}^+$ contains B, we know that $A \rightarrow B$ can be derived from M
  - Therefore, D can be removed from $AD \rightarrow B$
- New $M = \{A \rightarrow C, A \rightarrow D, A \rightarrow B\}$; done

3NF Decomposition Algorithm

- Given:
  - Table R(A, B, C, D)
  - A minimal basis {A→B, A→C, C→D}
- Step 1: Combine those FDs with the same left hand side
  - Result: {A→BC, C→D}
- Step 2: For each FD, create a table that contains all attributes in the FD
  - Result: R_1(A, B, C), R_2(C, D)
- Step 3: Remove redundant tables (if any)
- Tricky issue: Sometimes we also need to add an additional table (see the next slide)

3NF Decomposition Algorithm

- Given:
  - Table R(A, B, C, D)
  - A minimal basis {A→B, C→D}
- Step 1: Combine those FDs with the same left hand side
  - Result: {A→B, C→D}
- Step 2: For each FD, create a table that contains all attributes in the FD
  - Result: R_1(A, B), R_2(C, D)
- Step 3: Remove redundant tables (if any)
- Problem: R_1 and R_2 do not ensure lossless join
- Solution: Add a table that contains a key of the original table R
- Key of R: {A, C}
- Additional table to add: R_3(A, C)
- Final result: R_1(A, B), R_2(C, D), R_3(A, C)

3NF Decomposition Algorithm

- Given:
  - Table R(A, B, C, D)
  - A minimal basis {A→B, C→D}
- Step 1: Combine those FDs with the same left hand side
  - Result: {A→B, C→D}
- Step 2: For each FD, create a table that contains all attributes in the FD
  - Result: R_1(A, B), R_2(C, D)
- Step 3: If no table contain a key of the original table, add a table that contains a key of the original table
  - Result: R_1(A, B), R_2(C, D), R_3(A, C)
- Step 4: Remove redundant tables (if any)

This Lecture

- 3NF Decomposition
- Properties of 3NF

Why Does 3NF Decomposition Produce 3NF tables?

- Given: A table R, and a set S of FDs
- Step 1: Derive a minimal basis of S
- Step 2: In the minimal basis, combine the FDs whose left hand sides are the same
- Step 3: Create a table for each FD remained
- Step 4: If none of the tables contain a key of the original table R, create a table that contains a key of R
- Step 5: Remove redundant tables

Why Does 3NF Decomposition Produce 3NF tables?

- Given: A table R, and a set S of FDs
- Step 1: Derive a minimal basis of S
- Step 2: In the minimal basis, combine the FDs whose left hand sides are the same
- Step 3: Create a table for each FD remained
- Step 4: If none of the tables contain a key of the original table R, create a table that contains a key of R
- Step 5: Remove redundant tables
- Answer in Step 3

Why Does 3NF Decomposition Produce 3NF tables?

- Given: A table R, and a set S of FDs
- Step 1: Derive a minimal basis of S
- Step 2: In the minimal basis, combine the FDs whose left hand sides are the same
- Step 3: Create a table for each FD remained
- Example
  - Given FDs: A→B, B→C
  - Keys: {A}
  - A→B is OK and B→C is not OK
  - R_1(A, B) and R_2(B, C)

R

A	B	C

Minimal Basis is not always unique

- For given set of FDs, its minimal basis may not be unique
- Example:
  - Given R(A, B, C) and {A→B, A→C, B→C, B→A, C→A, C→B}
  - Minimal basis 1: {A→B, B→C, C→A}
  - Minimal basis 2: {A→C, B→C, C→A, C→B}
- Different minimal basis may lead to different 3NF decompositions

BCNF vs. 3NF

- BCNF: For any non-trivial FD
  - its left hand side (lhs) is a superkey
- 3NF: For any non-trivial FD
  - Either its lhs is a superkey
  - Or each attribute on its right hand side appear in a key
- Observation: BCNF is stricter than 3NF
- Therefore
  - A table that satisfies BCNF must satisfy 3NF, but not vice versa
  - A table that violates 3NF must violate BCNF, but not vice versa

BCNF vs. 3NF

- BCNF Decomposition:
  - Avoids insertion, deletion, and update anomalies
  - Eliminates most redundancies
  - But does not always preserve all FDs
- 3NF Decomposition:
  - Avoids insertion, deletion, and update anomalies
  - May lead to a bit more redundancy than BCNF
  - Always preserve all FDs
- So which one to use?
- A logical approach
  - Go for a BCNF decomposition first
  - If it preserves all FDs, then we are done
  - If not, then go for a 3NF decomposition instead

Why Does 3NF Preserve All FDs?

- Given: A table R, and a set S of FDs
- Step 1: Derive a minimal basis of S
- Step 2: In the minimal basis, combine the FDs whose left hand sides are the same
- Step 3: Create a table for each FD remained
- Step 4: If none of the tables contain a key of the original table R, create a table that contains a key of R
- Step 5: Remove redundant tables
- Rationale: Because of Step 3 (minimal basis preserves FDs; no redundant FDs)

Next lecture:

Topic 5: Relational Algebra (1)