Database Management Systems
Course 236363

Tutorial 8:
Decompositions and Normal Forms

Outline
- Decomposition
- Preserving information
  - Example
  - Algorithm for checking
- Preserving dependencies
  - Projecting FD's
- Normal Forms
  - BCNF
  - 3NF
- Questions

Decomposition

- A decomposition of R is a set \{R_1, \ldots, R_n\} such that \bigcup_{i=1}^n R_i = R

- Motivation:
  - Allows a better modeling of the database
- Characteristics of a good modeling:
  - Preserves information (necessary)
  - Preserves dependencies (not necessary, desired)

Preserving Information

- R - a relational schema
- F - a set of FD's
- P=\{R_1, \ldots, R_n\} decomposition
- P preserves information with respect to F if for every relation r over R such that r \not\models F the following holds:
  \[ \bigotimes_{i=1}^n \pi_{R_i}(r) = r \]

Preserving Information - example

- R=\{ID, Name, Address\}
- F=\{ID\rightarrow Name, ID\rightarrow Address\}
- Does the following decomposition preserve data?
  - P=\{R_1(ID, Name), R_2(Name, Address)
  - No!
Preserving Information – example (cont’d)

• Note that the relation obtained by joining the two projections is different than the original relation.

\[
\begin{array}{c|c|c|c}
\text{ID} & \text{NAME} & \text{ADDR} \\
1 & \text{Alice} & \text{CA} \\
2 & \text{Alice} & \text{TX} \\
\end{array}
\quad
\begin{array}{c|c|c|c}
\text{ID} & \text{NAME} & \text{ADDR} \\
1 & \text{Alice} & \text{CA} \\
2 & \text{Alice} & \text{TX} \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\text{ID} & \text{NAME} & \text{ADDR} \\
1 & \text{Alice} & \text{CA} \\
2 & \text{Alice} & \text{TX} \\
\end{array}
\]

\[
\begin{array}{c|c|c|c}
\text{ID} & \text{NAME} & \text{ADDR} \\
1 & \text{Alice} & \text{CA} \\
2 & \text{Alice} & \text{TX} \\
\end{array}
\]

Algorithm for information preserving

• R - a relational schema and \( P = \{R_1, \ldots, R_n\} \) a decomposition
• Does \( P \) preserves information?

- We show the algorithm’s run on the following example \( R(A, B, C, D, E, H) \)
- \( P = \{R_1(A, B), R_2(A, C, D), R_3(B, E, H)\} \)

Algorithm for information preserving

- In our example:

\[
\begin{array}{c|c|c|c|c|c|c|c|c|c}
A & B & C & D & E & H \\
\hline
1 & a & b & c & d & e & f \\
2 & a & c & d & e & f & h \\
3 & a & b & c_1 & d_1 & e_1 & h_1 \\
\end{array}
\]

\( R(A, B, C, D, E, H) \)

\( F = \{A \rightarrow B, C \rightarrow D, B \rightarrow EH\} \)

\( \rho = \{R_1(A, B), R_2(A, C, D), R_3(B, E, H)\} \)

Algorithm for information preserving

- 1st step - Initialization:
  - Create a relation \( r \) over \( R \) such that:
    - Every schema \( R_i \) has its own tuple \( t_i \)
    - For every attribute \( A \):
      - If \( A \in R_i \) then \( t_i[A] = a \)
      - Else \( t_i[A] = a_i \)
      - Similarly with \( b \) for \( B \), \( c \) for \( C \), etc.

Algorithm for information preserving

- 2nd step - chase
  - While the table changes do:
    - Look for an FD violation and equate the conclusions
    - “Equate” = change every occurrence of one to the other
      - When equating \( a_i \) with \( a \), change \( a_i \) with \( a \).

- 3rd step:
  - Return true if and only if there is a row without indexes.
Projection of FD's

- R - a relational schema
- F - a set of FD's
- $S \subseteq R$

- The projection of F on S, $\pi_S F$ is the set: 
  \[ \{ X \rightarrow Y | X \subseteq F \land X \cap Y \subseteq S \} \]

  - Intuitively, this is the set of FD's that are relevant to S

Outline

- Decomposition
- Preserving information
  - Example
  - Algorithm for checking
- Preserving dependencies
  - Projecting FD's
- Normal Forms
  - BCNF
  - 3NF
- Questions

Preserving Dependencies

- Intuition:
  - If each of the relations $r_i$ over $R_i$ satisfies the set of FD's that are relevant for $R_i$ (i.e., $\pi_{R_i} F_i$) then all of the database satisfies F.
- Goal: simple updates
  - If a decomposition does not preserve dependencies then when we update a relation we need to check whether this update is consistent with the other relations.
  - Therefore we can live without it (although it is desired).
Preserving Dependencies - example

- \( R(\text{Phone, AreaCode, City}) \)
- \( F=\{\text{City} \rightarrow \text{AreaCode}, \) \( \text{AreaCode}, \text{Phone} \) \( \rightarrow \) \( \text{City} \} \)
- \( P=\{R1(\text{Phone, City}), R2(\text{AreaCode, City})\} \)

- This decomposition preserves information:
  - \( \text{City} \) is in both \( R1 \) and \( R2 \)
  - \( R2 \) \( \setminus \) \( R1 \) = \( \text{AreaCode} \)
  - \( F \) implies \( \text{City} \rightarrow \text{AreaCode} \)

- Does it preserve dependencies?

Preserving Dependencies

- A decomposition preserves dependencies if all of \( F \)’s dependencies are preserved.
- A functional dependency \( f \) is preserved if
  - There exists a schema that includes all of the attributes in \( f \)
  - \( F \) can be deduced from other dependencies that are preserved in the decomposition

Outline

- Decomposition
- Preserving information
  - Example
  - Algorithm for checking
- Preserving dependencies
  - Projecting FD’s
- Normal Forms
  - BCNF
  - 3NF
- Questions

Normal Forms

- Normal Form is a characteristics of relational schema that captures the “quality” of the schema in the following sense:
  - A schema is better if it prevents duplications

- We will discuss the following Normal Forms (NF):
  - BCNF
  - 3NF
BCNF – Boyce-Codd NF

- A schema $R$ with a set $F$ of FD’s is in BCNF if every nontrivial FD implied by $F$ has a superkey on its premise (lhs)
  - That is, every $X \rightarrow Y$ in $F^+$ is such that $X$ is a superkey; or
  - $Y \subseteq X$

BCNF – example

- $R = \{\text{Id, Name, Address}\}$
- $F = \{\text{Id} \rightarrow \text{Name}, \text{Name} \rightarrow \text{Address}\}$

- $R$ is not BCNF w.r.t. $F$ since
  - The only key is Id
  - $\text{Name} \rightarrow \text{Address}$ does not satisfy the condition

- For $F' = \{\text{Id} \rightarrow \text{Name}, \text{Id} \rightarrow \text{Address}\}$
  - $R$ is BCNF w.r.t. $F'$

How to check whether a schema is BCNF?

By definition:
- Compute $F^+$
- For every FD $X \rightarrow Y \in F^+$ check whether $X$ is a superkey.

Problem: the size of $F^+$ is exponential in $R$

Theorem: If $R$ is not BCNF with respect to $F$ (there exists an FD $X \rightarrow Y \in F^+$ that violates the conditions), then there exists an FD $Z \rightarrow W \in F$ that violates the conditions.

BCNF Vs. Preserving Dependencies

- Often we need to choose between BCNF and Preserving Dependencies
  - How should we choose?
    - If we have lots of updates of attributes that have duplications in the original database (for instance AreaCode)
      - BCNF prevents duplications
    - If we want to add/update attributes that appear in an FD that is not preserved (for instance Phone)
      - We use 3NF

3NF

- $R$ – a relational schema
- $F$ – a set of FD’s over $R$

- $R$ is in 3NF if for every FD $X \rightarrow A \in F^*$ such that $A \notin X$
  - $X$ is a superkey of $R$ or
  - $A$ is contained in a key of $R$

3NF - Example

- $R(\text{City, AreaCode, Phone})$
- $F = \{\text{City} \rightarrow \text{AreaCode}, (\text{AreaCode, Phone}) \rightarrow \text{City}\}$

- 3NF
  - The keys are
    - $(\text{City, Phone})$, $(\text{AreaCode, Phone})$
  - Every FD from $F$ satisfies 3NF conditions
  - As in BCNF, it suffices to check only those FD’s in $F$
Normal Forms

- Every BCNF schema is also in 3NF
  - The opposite does not necessarily hold
- BCNF prevents more duplications than 3NF
- There always exists a 3NF decomposition that preserves dependencies and information
  - This is not true in BCNF and therefore we will sometimes prefer 3NF even though it is less efficient in sense of duplications

Algorithm for finding 3NF decomposition

- Given a minimal cover of FD's
  1. If there exists an FD in F that includes all of the attributes in R, return (R).
  2. For every set of FD's of the form $X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_n$, create a schema $(X, A_1, A_2, \ldots, A_n)$
  3. If none of the schemas contains a superkey of R, add a schema which is a superkey of R.

  - Note that this algorithm finds a decomposition that preserves information and dependencies.

3NF Decomposition - Example

$R(DName, DAddr, ID, PName, PAddr, PresNo, Date, MedName, Qnt)$

$F = \{ DName \rightarrow DAddr, ID \rightarrow PName, ID \rightarrow PAddr, ID \rightarrow DName, PresNo \rightarrow Date, PresNo \rightarrow ID, (PresNo, MedName) \rightarrow Qnt \}$

3NF Decomposition – Example (cont’d)

1. There is no FD that contains all of the attributes
2. We create the schemas:
   - $R_1(DName, DAddr)$
   - $R_2(ID, PName, PAddr, DName)$
   - $R_3(PresNo, Date, ID)$
   - $R_4(PresNo, MedName, Qnt)$
3. Note that $R_4$ contains the superkey $(PresNo, MedName)$ and therefore we do not need to add additional schema.

3NF

- Decompose the following schema to 3NF where:
  - $R(sid, sname, cnum, cname, grade)$
  - $F = \{ sid \rightarrow sname, cnum \rightarrow cname, (sid, cnum) \rightarrow grade \}$
- $R_1(sid, sname), R_2(cnum, cname), R_3(sid, cnu, grade)$

Outline

- Decomposition
- Preserving information
  - Example
  - Algorithm for checking
- Preserving dependencies
  - Projecting FD's
- Normal Forms
  - BCNF
  - 3NF
- Questions
Questions from Exam

R(A,B,C,D,E,H)
F={AB→H, E→BC, D→H, A→DE, C→E, D→BH}

Find a minimal cover to F
F is minimal if for every X→Y in F the following hold:
1. |Y| = 1
2. F → (X→Y)
3. For every Z ⊆ X it holds that F → (X→Y)

\[ G = (X→A) \land Y \land (X→Y) \land F \land (A→Y) \]
Repeat:
1. For each f = X→A ∈ G do
   i. If A X then G ← G ∪ (f)
2. For each f = X→Y ∈ G and B ∈ X do
   i. If Ap(f) = Y then G ← G ∪ (X→A) \cup (X→B) \cup (X→A) \cup (X→B)
   until no more changes to G

Questions from Exam

G={E→B, E→C, D→H, A→D, A→E, C→E, D→B}
ρ = \{R₁(A, B, D), R₂(A, C), R₃(C, D, E, H)\}

Is ρ in BCNF?

R₁ is not BCNF
π₁G ≠ \{A→B, A→D, D→B\}
D is not a superkey!

Questions from Exam

ρ = \{R₁(A, B, D), R₂(A, C), R₃(C, D, E, H)\}
G={E→B, E→C, D→H, A→D, A→E, C→E, D→B}

Does this decomposition preserve dependencies?

\[
\text{for each } f = (X→Y) \in \rho \text{ do begin }
\begin{align*}
& f \leftarrow X; \\
& \text{repeat}\text{ for } i = 1 \to n \\
& \quad \text{if } f \in \rho \\
& \quad \text{then } f \leftarrow \rho \text{ \& } f \leftarrow \rho \text{ \& } f \leftarrow \rho \text{ \& } f \leftarrow \rho \\
& \quad \text{until no more change in } f.
\end{align*}
\]
And ρ is dependency preserving iff all \((X→Y) \in \rho\) are preserved.
Questions from Exam

\[ p = \{R_1(A, B, D), R_2(A, C), R_3(C, D, E, H)\} \]
\[ G = \{E \rightarrow B, E \rightarrow C, D \rightarrow H, A \rightarrow D, A \rightarrow E, C \rightarrow E, D \rightarrow B\} \]

E \rightarrow B is not contained in any schema

- \( Z = \{E\} \)
- \( Z \cap R_1 = \{\} \) no change to \( Z \)
- \( Z \cap R_2 = \{\} \) no change to \( Z \)
- \( Z \cap R_3 = \{E\} \)
  - \( \{B\} \cap R_1 \cap \{C, E\} \)
  - \( Z \cap \{C, E\} \)
- \( Z \cap R_1 = \{\} \) no change to \( Z \)
- \( Z \cap R_2 = \{\} \) no change to \( Z \)

E \rightarrow B is not preserved!