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)
Outline

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

\[ \forall_{i=1\ldots n} \pi_{R_i}(r) = r \]
Preserving Information - example

- \( R = \{ \text{ID, Name, Address} \} \)
- \( F = \{ \text{ID} \rightarrow \text{Name, ID} \rightarrow \text{Address} \} \)
- Does the following decomposition preserve data?
  - \( P = \{ R_1(\text{ID, Name}), R_2(\text{Name, Address}) \} \)

- No!
• Note that the relation obtained by joining the two projections is different than the original relation.
Preserving Information – example

- However, the decomposition
  \[ \rho' = \{R'_1(ID, NAME), R'_2(ID, ADDR)\} \]
  preserves information.
- Indeed, \( \pi_{R'_1}(r) \bowtie \pi_{R'_2}(r) = r \)

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

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

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

- 1\textsuperscript{st} 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 \( t[A] = a \)
      - Else \( t[A] = a_i \)
      - Similarly with \( b \) for \( B \), \( c \) for \( C \), etc.
Algorithm for information preserving

• In our example:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>a</td>
<td>b</td>
<td>c₁</td>
<td>d₁</td>
<td>e₁</td>
<td>h₁</td>
</tr>
<tr>
<td>t₂</td>
<td>a</td>
<td>b₂</td>
<td>c</td>
<td>d</td>
<td>e₂</td>
<td>h₂</td>
</tr>
<tr>
<td>t₃</td>
<td>a₃</td>
<td>b</td>
<td>c₃</td>
<td>d₃</td>
<td>e</td>
<td>h</td>
</tr>
</tbody>
</table>

R(A, B, C, D, E, H)
F = {A → B, C → D, B → EH}
ρ = {R₁(A, B), R₂(A, C, D), R₃(B, E, H)}
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 $a_j$ with $a$, change $a_j$ with $a$.

• 3nd step:
  – Return true if and only if there is a row without indexes.
Algorithm for information preserving

\[ F = \{A \rightarrow B, C \rightarrow D, B \rightarrow EH\} \]

\[ A \rightarrow B (t_1, t_2) \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>a</td>
<td>b</td>
<td>c₁</td>
<td>d₁</td>
<td>e₁</td>
<td>h₁</td>
</tr>
<tr>
<td>t₂</td>
<td>a</td>
<td>b₂</td>
<td>c</td>
<td>d</td>
<td>e₂</td>
<td>h₂</td>
</tr>
<tr>
<td>t₃</td>
<td>a₃</td>
<td>b</td>
<td>c₃</td>
<td>d₃</td>
<td>e</td>
<td>h</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>a</td>
<td>b</td>
<td>c₁</td>
<td>d₁</td>
<td>e₁</td>
<td>h₁</td>
</tr>
<tr>
<td>t₂</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e₂</td>
<td>h₂</td>
</tr>
<tr>
<td>t₃</td>
<td>a₃</td>
<td>b</td>
<td>c₃</td>
<td>d₃</td>
<td>e</td>
<td>h</td>
</tr>
</tbody>
</table>
Algorithm for information preserving

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>a</td>
<td>b</td>
<td>c₁</td>
<td>d₁</td>
<td>e₁</td>
<td>h₁</td>
</tr>
<tr>
<td>t₂</td>
<td>a</td>
<td>b₂</td>
<td>c</td>
<td>d</td>
<td>e₂</td>
<td>h₂</td>
</tr>
<tr>
<td>t₃</td>
<td>a₃</td>
<td>b</td>
<td>c₃</td>
<td>d₃</td>
<td>e</td>
<td>h</td>
</tr>
</tbody>
</table>

F = {A → B, C → D, B → EH}

1. A → B(t₁,t₂)

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>t₁</td>
<td>a</td>
<td>b</td>
<td>c₁</td>
<td>d₁</td>
<td>e₁</td>
<td>h₁</td>
</tr>
<tr>
<td>t₂</td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e₂</td>
<td>h₂</td>
</tr>
<tr>
<td>t₃</td>
<td>a₃</td>
<td>b</td>
<td>c₃</td>
<td>d₃</td>
<td>e</td>
<td>h</td>
</tr>
</tbody>
</table>

2. B → EH(t₂,t₃)

- Note that we have a tuple without subscripts and thus the decomposition preserves information
Information Preserving - example

- \( R(A,B,C,D,E) \)
- \( F = \{ A \rightarrow B, \ B \rightarrow C, \ C \rightarrow D, \ DE \rightarrow BC \} \)
- \( \rho = \{ R1(A,D), R2(A,E), R3(B,C,D,E) \} \)
The Case of Binary Decomposition

THEOREM: Let \( \{X_1, X_2\} \) be a decomposition of \((U,F)\). The following are equivalent:

1. \( F \models (X_1 \cap X_2) \rightarrow X_1 \) or \( F \models (X_1 \cap X_2) \rightarrow X_2 \)
2. \( \{X_1, X_2\} \) is a lossless decomposition

So what would be a decision algorithm in this case?
• Decomposition
• Preserving information
  – Example
  – Algorithm for checking
• Preserving dependencies
  – Projecting FD’s
• Normal Forms
  – BCNF
  – 3NF
• Questions
• 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:
  \[
  \{ X \rightarrow Y \mid X \rightarrow Y \in F^+ \land X \cup Y \subseteq S \}.
  \]

  – Intuitively, this is the set of FD’s that are relevant to S
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

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

• This decomposition preserves information:
  – City is in both \( R1 \) and \( R2 \)
  – \( R2 \backslash R1 = \text{AreaCode} \)
  – \( F \) implies \( \text{City} \rightarrow \text{AreaCode} \)
• Does it preserve dependencies?
Preserving Dependencies - example

- **R(Phone, AreaCode, City)**
- **F={City \rightarrow AreaCode, (AreaCode, Phone) \rightarrow City}**
- **P={R1(Phone, City), R2(AreaCode, City)}**
- Does this decomposition preserve dependencies?
  - No!
  - The FD [(AreaCode, Phone) \rightarrow City] is not preserved
Preserving Dependencies

• A decomposition preserves dependencies if all of F’s dependencies are preserved.

• A functional dependency $f$ is preserved if
  – There is a schema $X$ where $f$ in $F[X]$
  – $f$ can be deduced from other dependencies that are preserved in the decomposition

• In other words, $F^+ = (F [X1] \cup \cdots \cup F [Xk])^+$
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 \( \text{Id} \)
  - \( \text{Name} \rightarrow \text{Address} \) 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.
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
    - (City, Phone), (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 3NF Decomposition

3NFDec(U,F) {
    D = ∅
    G := MinCover(F)
    for all (X→A in G) do
        D := D∪{XA}
        if (no set in D is a superkey)
            D := D∪{FindKey(U,F)}
    D := RemoveConained(D)
    return D
}
3NF Decomposition - Example

R(DName, Daddr, ID, PAddr, PName, PresNo, MedName, Qnt)

F = \{ DName \rightarrow DAddr, 
     ID \rightarrow PName, 
     (ID, PName) \rightarrow PAddr, 
     PresNo \rightarrow ID, 
     (PresNo, MedName) \rightarrow Qnt \}
1. Minimal cover:
   \[F = \{ \text{DName} \rightarrow \text{DAddr},\]
   \[\text{ID} \rightarrow \text{PName},\]
   \[\text{ID} \rightarrow \text{PAddr},\]
   \[\text{PresNo} \rightarrow \text{ID},\]
   \[(\text{PresNo}, \text{MedName}) \rightarrow \text{Qnt} \} \]

2. We create the schemas:
   \[R_1(\text{DName}, \text{DAddr})\]
   \[R_2(\text{ID}, \text{PName})\]
   \[R_3(\text{ID}, \text{PAddr})\]
   \[R_4(\text{PresNo}, \text{ID})\]
   \[R_5(\text{PresNo}, \text{MedName}, \text{Qnt})\]

3. Adding a key : \(R_6(\text{PresNo}, \text{MedName}, \text{DName})\)

4. No contained schemas to remove
Final Decomposition:

\[ R_1(\text{DName}, \text{DAddr}) \]
\[ R_2(\text{ID}, \text{PName}) \]
\[ R_3(\text{ID}, \text{PAddr}) \]
\[ R_4(\text{PresNo}, \text{ID}) \]
\[ R_5(\text{PresNo}, \text{MedName}, \text{Qnt}) \]
\[ R_6(\text{PresNo}, \text{MedName}, \text{DName}) \]
• Decompose the following schema to 3NF where:
  – $R(sid, sname, cnum, cnum, cnum, grade)$
  – $F = \{ \text{sid} \rightarrow \text{sname}, \text{cnum} \rightarrow \text{name},$
    \hspace{1cm} (\text{sid}, \text{cnum}) \rightarrow \text{grade} \}$

• $R_1(sid, sname), R_2(\text{cnum}, \text{name}), R_3(\text{sid, cnum, grade})$
Outline

- Decomposition
- Preserving information
  - Example
  - Algorithm for checking
- Preserving dependencies
  - Projecting FD’s
- Normal Forms
  - BCNF
  - 3NF
- Questions
R(A,B,C,D,E,H)
F={AB \rightarrow H, E \rightarrow BC, D \rightarrow H, A \rightarrow DE, C \rightarrow E, D \rightarrow BH}

**Find a minimal cover to F**
F is minimal if for every X\rightarrow Y in F the following hold:
1. \(|Y| = 1\)
2. F^+ \neq (F \setminus \{X \rightarrow Y\})^+
3. For every Z \subseteq X it holds that F^+ \neq ((F \setminus \{X \rightarrow Y\}) \cup \{Z \rightarrow Y\})^+

G ← \{(X \rightarrow A) \mid \exists Y ((X \rightarrow Y) \in F \land A \in Y)\};
Repeat
1. For each f = X \rightarrow A \in G do
   if A \in X^+_{G \setminus \{f\}} then G ← G \setminus \{f\};
2. For each f = X \rightarrow A \in G and B \in X do
   if A \in (X\{B\})^+_G then G ← (G\{X \rightarrow A\}) \cup \{X\{B\} \rightarrow A\};
until no more changes to G
Questions from Exam

• Initialization
  \[ G = \{ AB \rightarrow H, E \rightarrow B, E \rightarrow C, A \rightarrow D, A \rightarrow E, C \rightarrow E, D \rightarrow B, D \rightarrow H \} \]

• Step 1:
  – We omit the FD \( AB \rightarrow H \) since \( H \in G \setminus \{AB \rightarrow H\}^+ \)
  – \( G = \{ E \rightarrow B, E \rightarrow C, D \rightarrow H, A \rightarrow D, A \rightarrow E, C \rightarrow E, D \rightarrow B, D \rightarrow H \} \)

• Step 2: no change
• Step 1: no change
• G={E →B, E →C, D →H, A →D, A →E, C →E, D →B}
Questions from Exam

\[ G = \{ E \rightarrow B, \ E \rightarrow C, \ D \rightarrow H, \ A \rightarrow D, \ A \rightarrow E, \ C \rightarrow E, \ D \rightarrow B \} \]

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

Is \( \rho \) in BCNF?

\( R_1 \) is not BCNF

\( \pi_{R_1}G = \{ A \rightarrow B, \ A \rightarrow D, \ D \rightarrow B \} \)

\( D \) is not a superkey!
Questions from Exam

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

Does this decomposition preserve information?
### Questions from Exam

G = \{E \rightarrow B, \ E \rightarrow C, \ D \rightarrow H, \ A \rightarrow D, \ A \rightarrow E, \ C \rightarrow E, \ D \rightarrow B\}

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c₁</td>
<td>d</td>
<td>e₁</td>
<td>h</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>h₂</td>
<td></td>
</tr>
<tr>
<td>a₃</td>
<td>b₃</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>h</td>
<td></td>
</tr>
</tbody>
</table>

\[ E \rightarrow B \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c₁</td>
<td>d</td>
<td>e₁</td>
<td>h</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>h₂</td>
<td></td>
</tr>
<tr>
<td>a₃</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>h</td>
<td></td>
</tr>
</tbody>
</table>

Yes!

\[ D \rightarrow H \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b</td>
<td>c₁</td>
<td>d</td>
<td>e</td>
<td>h</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>h₂</td>
<td></td>
</tr>
<tr>
<td>a₃</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e</td>
<td>h</td>
<td></td>
</tr>
</tbody>
</table>

236363 - DBMS, Design
Questions from Exam

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

**Does this decomposition preserve dependencies?**

for each \( f = (X \rightarrow Y) \in F \) do begin

\( Z_f \leftarrow X; \)

repeat

for \( i = 1 \) to \( n \) do

\( Z_f \leftarrow Z_f \cup ((Z_f \cap R_i)^+_F \cap R_i) \)

until no more change to \( Z_f \)

\( X \rightarrow Y \) is preserved iff \( Y \subseteq Z_f \)

end;

\( \rho \) is dependency preserving iff all \( (X \rightarrow Y) \in F \) are preserved
Questions from Exam

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

\(E \rightarrow B\) is not contained in any schema

- \(Z_f = \{E\}\)
- \(Z_f \cap R_1 = \{\}\) no change to \(Z_f\)
- \(Z_f \cap R_2 = \{\}\) no change to \(Z_f\)
- \(Z_f \cap R_3 = \{E\}\)
  \(\{E\}^+ \cap R_3 = \{C, E\}\)
  \(\Rightarrow Z_f = \{C, E\}\)

- \(Z_f \cap R_1 = \{\}\) no change to \(Z_f\)
- \(Z_f \cap R_2 = \{\}\) no change to \(Z_f\)

\(E \rightarrow B\) is not preserved!