Tutorial 8:

Decompositions and Normal Forms
• 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:

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

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

- No!
Preserving Information – example (cont’d)

Note that the relation obtained by joining the two projections is different than the original relation.
Preserving Information – example

• However, the decomposition
  \( \rho' = \{ R'_1(\text{ID}, \text{NAME}), R'_2(\text{ID}, \text{ADDR}) \} \)
  preserves information.

• Indeed, \( \pi_{R'_1}(r) \bowtie \pi_{R'_2}(r) = r \)

\[ r = \]

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
<th>ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
<td>CA</td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
<td>TX</td>
</tr>
</tbody>
</table>

\[ \pi_{R'_1}(r) \]

<table>
<thead>
<tr>
<th>ID</th>
<th>NAME</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Alice</td>
</tr>
<tr>
<td>2</td>
<td>Alice</td>
</tr>
</tbody>
</table>

\[ \pi_{R'_2}(r) \]

<table>
<thead>
<tr>
<th>ID</th>
<th>ADDR</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>CA</td>
</tr>
<tr>
<td>2</td>
<td>TX</td>
</tr>
</tbody>
</table>
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 \( R(A, B, C, D, E, H) \)
- \( P = \{R1(A, B), R2(A, C, D), R3(B, E, H)\} \)
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.
• 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>$t_1$</td>
<td>a</td>
<td>b</td>
<td>$c_1$</td>
<td>$d_1$</td>
<td>$e_1$</td>
<td>$h_1$</td>
</tr>
<tr>
<td>$t_2$</td>
<td>a</td>
<td>$b_2$</td>
<td>c</td>
<td>d</td>
<td>$e_2$</td>
<td>$h_2$</td>
</tr>
<tr>
<td>$t_3$</td>
<td>$a_3$</td>
<td>b</td>
<td>$c_3$</td>
<td>$d_3$</td>
<td>e</td>
<td>h</td>
</tr>
</tbody>
</table>

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

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

$\rho = \{R_1(A, B), R_2(A, C, D), R_3(B, E, H)\}$
Algorithm for information preserving

• 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><strong>t₁</strong></td>
<td>a</td>
<td>b</td>
<td>c₁</td>
<td>d₁</td>
<td>e₁</td>
<td>h₁</td>
</tr>
<tr>
<td><strong>t₂</strong></td>
<td>a</td>
<td>b₂</td>
<td>c</td>
<td>d</td>
<td>e₂</td>
<td>h₂</td>
</tr>
<tr>
<td><strong>t₃</strong></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><strong>t₁</strong></td>
<td>a</td>
<td>b</td>
<td>c₁</td>
<td>d₁</td>
<td>e₁</td>
<td>h₁</td>
</tr>
<tr>
<td><strong>t₂</strong></td>
<td>a</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td>e₂</td>
<td>h₂</td>
</tr>
<tr>
<td><strong>t₃</strong></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

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

- \( R(A,B,C,D,E) \)
- \( F = \{A \rightarrow B, B \rightarrow C, C \rightarrow D, DE \rightarrow BC\} \)
- \( \rho = \{R1(A,D), R2(A,E), R3(B,C,D,E)\} \)
Outline

• Decomposition
• Preserving information
  – Example
  – Algorithm for checking
• Preserving dependencies
  – Projecting FD’s
• Normal Forms
  – BCNF
  – 3NF
• Questions
Projection of FD’s

- R - a relational schema
- F – a set of FD’s
- S ⊆ 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, $\pi_{R_i} F$,) 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

- \(R(\text{Phone, AreaCode, City})\)
- \(F=\{\text{City} \rightarrow \text{AreaCode}, \ (\text{AreaCode, Phone}) \rightarrow \text{City}\}\)
- \(P=\{R_1(\text{Phone, City}), R_2(\text{AreaCode, City})\}\)

- This decomposition preserves information:
  - City is in both \(R_1\) and \(R_2\)
  - \(R_2\setminus R_1=\text{AreaCode}\)
  - \(F\) implies \(\text{City} \rightarrow \text{AreaCode}\)

- Does it preserve dependencies?
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?*
Preserving Dependencies - example

- R(Phone, AreaCode, City)
- F={City → AreaCode, (AreaCode, Phone) → City}
- P={R1(Phone, City), R2(AreaCode, City)}
- Does this decomposition preserve dependencies?
  - No!
  - The FD [(AreaCode, Phone) → 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 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 characteristic 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
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

• \( R = \{\text{Id}, \text{Name}, \text{Address}\} \)
• \( F=\{\text{Id} \rightarrow \text{Name}, \text{Name} \rightarrow \text{Address}\} \)

• \( R \) 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.
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(City, AreaCode, Phone)**
- **F={City→AreaCode, (AreaCode, Phone)→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 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 $\{X, A_1, \ldots, A_n\}$.
  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\}
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.
• Decompose the following schema to 3NF where:
  – R(sid, sname, cnum, cname, grade)
  – F = {sid → sname, cnum → cname,
        (sid, cnum) → grade}

• R₁(sid, sname), R₂(cnum, cname),
  R₃(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⁺ ≠ (F \ {X → Y})⁺
3. For every Z ⊂ X it holds that F⁺ ≠ ((F \ {X → Y}) ∪ {Z → Y})⁺

G ← {(X→ A) | ∃Y ((X →Y) ∈ F ∧ A ∈ Y)};
Repeat
1. For each f = X → A ∈ G do
   if A ∈ X⁺ _G \ {f} then G ← G \ {f};
2. For each f = X → A ∈ G and B ∈ X do
   if A ∈ (X \ {B})⁺ _G then G ←(G \ {X → A}) ∪ {X \ {B} → A};
until no more changes to G
Questions from Exam

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

- **Step 1:**
  - We omit the FD \( \text{AB} \rightarrow \text{H} \) since \( \text{H} \in \text{AB}^+_{G\setminus\{\text{AB} \rightarrow \text{H}\}} \)
  - \[ G = \{ \text{E} \rightarrow \text{B}, \text{E} \rightarrow \text{C}, \text{D} \rightarrow \text{H}, \text{A} \rightarrow \text{D}, \text{A} \rightarrow \text{E}, \text{C} \rightarrow \text{E}, \text{D} \rightarrow \text{B}, \text{D} \rightarrow \text{H} \} \]

- **Step 2:** no change

- **Step 1:** no change

- **Step 1:** no change

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

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

**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

\[ Z_f \leftarrow Z_f \cup ((Z_f \cap R_f)^+ \cap R_f) \]

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
ρ = \{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 \(Z_f\)
• \(Z_f \cap R_3 = \{E\}\)
  \(\{E\}^+ \cap R_1 = \{C, E\}\)
  \(\Rightarrow Z_f = \{C, E\}\)

E \rightarrow B is not preserved!