ICS 321 Data Storage & Retrieval

Normal Forms (ii)

Asst. Prof. Lipyeow Lim
Information & Computer Science Department
University of Hawaii at Manoa
### Redundancies & Decompositions

<table>
<thead>
<tr>
<th>SSN</th>
<th>Name</th>
<th>Lot</th>
<th>Rating</th>
<th>Hourly_wages</th>
<th>Hours_worked</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-2366</td>
<td>Attishoo</td>
<td>48</td>
<td>8</td>
<td>10</td>
<td>40</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>10</td>
<td>30</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>7</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>7</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>10</td>
<td>40</td>
</tr>
</tbody>
</table>

### RatingWages

<table>
<thead>
<tr>
<th>Rating</th>
<th>Hourly_wages</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>7</td>
</tr>
<tr>
<td>8</td>
<td>10</td>
</tr>
</tbody>
</table>
Lossless-join Decomposition

• Decomposition of R into X and Y is lossless-join w.r.t. a set of FDs F if, for every instance r that satisfies F:

$$\pi_X(r) \text{ join } \pi_Y(r) = r$$

• In general one direction $\pi_X(r) \text{ join } \pi_Y(r) \subseteq r$ is always true, but the other may not hold.

• Definition extended to decomposition into 3 or more relations in a straightforward way.

• It is essential that all decompositions used to deal with redundancy be lossless! (Avoids Problem (2).)
Conditions for Lossless Join

• The decomposition of R into X and Y is lossless-join wrt F if and only if the closure of F contains:
  – \( X \cap Y \rightarrow X \), or
  – \( X \cap Y \rightarrow Y \)

• In particular, the decomposition of R into UV and \( R - V \) is lossless-join if \( U \rightarrow V \) holds over R.
Chase Test for Lossless Join

- R(A,B,C,D) is decomposed into S1={A,D}, S2={A,C}, S3={B,C,D}
- Construct a Tableau
  - One row for each decomposed relation
  - For each row i, subscript an attribute with i if it does not occur in Si.
- FDs: A→B, B→C, CD→A
- Rules for “equating two rows” using FDs:
  - If one is unsubscripted, make the other the same
  - If both are subscripted, make the subscripts the same
- Goal: one unsubscripted row
Dependency-preserving Decomposition

- **Dependency preserving decomposition (Intuitive):**
  - If R is decomposed into X, Y and Z, and we enforce the FDs that hold on X, on Y and on Z, then all FDs that were given to hold on R must also hold. *(Avoids Problem (3).)*

- **Projection of set of FDs F:** If R is decomposed into X, ... projection of F onto X (denoted $F_X$) is the set of FDs $U \rightarrow V$ in $F^+$ (*closure of F*) such that $U, V$ are in X.
Dependency-preserving Decomp. (Cont)

Decomposition of R into X and Y is **dependency preserving** if \((F_X \cup F_Y)^+ = F^+\)

- If we 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 \(F^+\).
- Example: ABC decomposed into AB and BC.
  - \(F=\{A \rightarrow B, \ B \rightarrow C, \ C \rightarrow A\}\).
  - Is this dependency preserving?
- Dependency preserving does not imply lossless join:
  - Eg. ABC, A \(\rightarrow\) B, decomposed into AB and BC.
  - And vice-versa! (Example?)

Important: \(F^+\), not \(F\)
Decomposition into BCNF

• Consider relation R with FDs F. \textit{How do we decompose R into a set of small relations that are in BCNF?}

• Algorithm:
  – If $X \rightarrow Y$ violates BCNF, decompose R into $R-Y$ and $XY$
  – Repeat until all relations are in BCNF.

• Example: $ABCD$, $B \rightarrow C$, $C \rightarrow D$, $C \rightarrow A$.

• Order in which we deal with the violating FD can lead to different relations!
BCNF Decomposition Algorithm (3.20)

- **Input**: $R_0$, set of FDs $S_0$
- **Output**: A decomposition of $R_0$ into a collection of relations, all of which are in BCNF
- Initially $R = R_0$, $S = S_0$

1. If $R$ is in BCNF, return \{R\}
2. Let $X \rightarrow Y$ be a violation.
   a. Compute $X+$.
   b. Choose $R_1 = X+$
   c. Let $R_2 = X$ union $(R - X+)$
3. Compute FD projections $S_1$ and $S_2$ for $R_1$ and $R_2$
4. Recursively decompose $R_1$ and $R_2$ and return the union of the results
BCNF & Dependency Preservation

• BCNF decomposition using Algo 3.20 is lossless join

• BUT in general there may not be a dependency preserving decomposition into BCNF

  – Example: Bookings(Title, City, Theater), with FD’s : Th→C, TiC→Th
  – Not in BCNF.
  – Can’t decompose while preserving 2nd FD;

• This is the motivation for 3NF.
Decomposition into 3NF

• Obviously, the algorithm for lossless join decomp into BCNF can be used to obtain a lossless join decomp into 3NF (typically, can stop earlier).

• How can we ensure dependency preservation?
  – If $X \rightarrow Y$ is not preserved, add relation $XY$.
  – Problem is that $XY$ may violate 3NF!
  – Example: $JP \rightarrow C$ is not preserved, so add $JPC$. What if FDs also include $J \rightarrow C$?

• Refinement: Instead of the given set of FDs $F$, use a minimal cover for $F$. 

Lipyew Lim -- University of Hawaii at Manoa


Minimum Cover for a Set of FDs

• **Minimal cover or basis** $G$ for a set of FDs $F$:
  – Closure of $F = \text{closure of } G$.
  – Right hand side of each FD in $G$ is a single attribute.
  – If we modify $G$ by deleting an FD or by deleting attributes from an FD in $G$, the closure changes.

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

• e.g., $A \rightarrow B$, $ABCD \rightarrow E$, $EF \rightarrow GH$, $ACDF \rightarrow EG$ has the following minimal cover:
  – $A \rightarrow B$, $ACD \rightarrow E$, $EF \rightarrow G$ and $EF \rightarrow H$
Computing the Minimal Cover

• Algorithm
  1. Put the FDs into standard form \( X \rightarrow A \). RHS is a single attribute.
  2. Minimize the LHS of each FD. For each FD, check if we can delete an attribute from LHS while preserving \( F^+ \).
  3. Delete redundant FDs.

• Minimal covers are not unique. Different order of computation can give different covers.
• e.g., \( A \rightarrow B, \ AB \rightarrow E, \ EF \rightarrow GH, \ ACDF \rightarrow EG \) has the following minimal cover:
  – \( A \rightarrow B, \ ACD \rightarrow E, \ EF \rightarrow G \) and \( EF \rightarrow H \)
3NF Decomposition Algorithm (3.26)

• **Input**: R, set of FDs F
• **Output**: A decomposition of R into a collection of relations, all of which are in BCNF

1. Find a minimal basis/cover for F, say G
2. For each FD $X \rightarrow A$ in G, use $XA$ as one of the decomposed relations.
3. If none of the relations from Step 2 is a superkey for R, add another relation whose schema is a key for R.
Summary of Schema Refinement

• If a relation is in BCNF, it is free of redundancies that can be detected using FDs. Thus, trying to ensure that all relations are in BCNF is a good heuristic

• If a relation is not in BCNF, we can try to decompose it into a collection of BCNF relations.
  -- Must consider whether all FDs are preserved.
  -- If a lossless-join, dependency preserving decomposition into BCNF is not possible (or unsuitable, given typical queries), should consider decomposition into 3NF.
  -- Decompositions should be carried out and/or re-examined while keeping performance requirements in mind.