Schema Refinement & Normalization Theory 3

Week 12 - 1
Normal Forms

• The first question: Is any refinement needed?

• Normal forms:
  – If a relation is in a certain normal form (BCNF, 3NF etc.), it is known that certain kinds of problems are avoided/minimized. This can be used to help us decide whether decomposing the relation will help.

• Role of FDs in detecting redundancy:
  – Consider a relation R with 3 attributes, ABC.
    • No FDs hold: There is no redundancy here.
    • Given A → B: Several tuples could have the same A value, and if so, they’ll all have the same B value!
Normal Forms

• First normal form (1NF)
  – Every field must contain atomic values, i.e. no sets or lists.
  – Essentially all relations are in this normal form

• Second normal form (2NF)
  – Any relation in 2NF is also in 1NF
  – All the non-key attributes must depend upon the WHOLE of the candidate key rather than just a part of it.
    • It is only relevant when the key is composite, i.e., consists of several fields.
  – e.g. Consider a relation:
    • Inventory(part, warehouse, quantity, warehouse_address).
    • Suppose {part, warehouse} is a candidate key.
    • warehouse_address depends upon warehouse alone - 2NF violation
    • Solution: decompose
Normal Forms

- Boyce-Codd Normal Form (BCNF)
  - Any relation in BCNF is also in 2NF

- Third normal form (3NF)
  - Any relation in BCNF is also in 3NF
Boyce-Codd Normal Form (BCNF)

- Reln R with FDs $F$ is in BCNF if for each non-trivial FD $X \rightarrow A$ in $F$, $X$ is a super key for $R$ (i.e., $X \rightarrow R$ in $F^+$).
  - An FD $X \rightarrow A$ is said to be “trivial” if $A \in X$.
- In other words, R is in BCNF if the only non-trivial FDs that hold over R are key constraints.
- If BCNF:
  - No “data” in R can be predicted using FDs alone. Why:
    - Because $X$ is a (super)key, we can’t have two different tuples that agree on the $X$ value.

Suppose we know that this instance satisfies $X \rightarrow A$. This situation cannot arise if the relation is in BCNF.
Decomposition of a Relation Schema

• When a relation schema is not in BCNF: \textbf{decompose}.
• Suppose that relation R contains attributes $A_1 \ldots A_n$. A \textit{decomposition} of R consists of replacing R by two or more relations such that:
  – Each new relation scheme contains a subset of the attributes of R (and no attributes that do not appear in R), and
  – Every attribute of R appears as an attribute of at least one of the new relations.
• Intuitively, decomposing R means we will store instances of the relation schemes produced by the decomposition, instead of instances of R.
Decomposition example

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>W</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</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>

Original relation (not stored in DB!)

Decomposition (in the DB)

```
<table>
<thead>
<tr>
<th>R</th>
<th>W</th>
</tr>
</thead>
<tbody>
<tr>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>5</td>
<td>7</td>
</tr>
</tbody>
</table>
```
Problems with Decompositions

• There are three potential problems to consider:
  ① Some queries become more expensive.
    • e.g., How much did sailor Attishoo earn? (earn = W*H)
  ② Given instances of the decomposed relations, we may not be able to reconstruct the corresponding instance of the original relation!
    • Fortunately, not in the SNLRWH example.
  ③ Checking some dependencies may require joining the instances of the decomposed relations.
    • Fortunately, not in the SNLRWH example.

• Tradeoff: Must consider these issues vs. redundancy.
Example of problem 2

<table>
<thead>
<tr>
<th>Student_ID</th>
<th>Name</th>
<th>Dcode</th>
<th>Cno</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Attishoo</td>
<td>INFS</td>
<td>501</td>
<td>A</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Guldu</td>
<td>CS</td>
<td>102</td>
<td>B</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>INFS</td>
<td>614</td>
<td>B</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>INFS</td>
<td>614</td>
<td>A</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>INFS</td>
<td>612</td>
<td>C</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Dcode</th>
<th>Cno</th>
<th>Grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>Attishoo</td>
<td>INFS</td>
<td>501</td>
<td>A</td>
</tr>
<tr>
<td>Guldu</td>
<td>CS</td>
<td>102</td>
<td>B</td>
</tr>
<tr>
<td>Smethurst</td>
<td>INFS</td>
<td>614</td>
<td>B</td>
</tr>
<tr>
<td>Guldu</td>
<td>INFS</td>
<td>614</td>
<td>A</td>
</tr>
<tr>
<td>Guldu</td>
<td>INFS</td>
<td>612</td>
<td>C</td>
</tr>
</tbody>
</table>
Lossless Join Decompositions

• Decomposition of R into $R_1$ and $R_2$ is \textit{lossless-join} w.r.t. a set of FDs F if, for every instance $r$ that satisfies F, we have:
\[
\pi_{R_1}(r) \bowtie \pi_{R_2}(r) = r
\]

• It is always true that
\[
r \subseteq \pi_{R_1}(r) \bowtie \pi_{R_2}(r)
\]

• In general, the other direction does not hold! If it does, the decomposition is \textit{lossless-join}.
Example (lossy decomposition)

\[
\begin{array}{|c|c|c|}
\hline
A & B & C \\
\hline
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 2 & 8 \\
\hline
\end{array}
\]

\[\pi_{AB}(r)\]

\[
\begin{array}{|c|c|}
\hline
A & B \\
\hline
1 & 2 \\
4 & 5 \\
7 & 2 \\
\hline
\end{array}
\]

\[\pi_{AB}(r) \bowtie \pi_{BC}(r)\]

\[
\begin{array}{|c|c|c|}
\hline
A & B & C \\
\hline
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 2 & 8 \\
1 & 2 & 8 \\
7 & 2 & 3 \\
\hline
\end{array}
\]
Example (lossless join decomposition)

\[ r \]

\[
\begin{array}{ccc}
A & B & C \\
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 2 & 3 \\
\end{array}
\]

\[ \pi_{AB}(r) \]

\[
\begin{array}{cc}
A & B \\
1 & 2 \\
4 & 5 \\
7 & 2 \\
\end{array}
\]

\[ \pi_{BC}(r) \]

\[
\begin{array}{cc}
B & C \\
2 & 3 \\
5 & 6 \\
\end{array}
\]

\[ \pi_{AB}(r) \bowtie \pi_{BC}(r) \]

\[
\begin{array}{ccc}
A & B & C \\
1 & 2 & 3 \\
4 & 5 & 6 \\
7 & 2 & 3 \\
\end{array}
\]

We have \((AB \cap BC) \rightarrow BC\)
Lossless Join Decomposition

• The decomposition of $R$ into $R_1$ and $R_2$ is lossless-join wrt $F$ if and only if $F^+$ contains:
  - $R_1 \cap R_2 \rightarrow R_1$, or
  - $R_1 \cap R_2 \rightarrow R_2$

• In particular, the decomposition of $R$ into $(UV)$ and $(R-V)$ is lossless-join if $U \rightarrow V$ holds on $R$
  - assume $U$ and $V$ do not share attributes.
  - WHY?
Decomposition

• 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))
Decomposition into BCNF

• Recall that for $X \rightarrow A$ in $F$ over $R$ to satisfy BCNF requirement, one of the followings must be true:
  – $XA$ are not all in $R$, or
  – $X \rightarrow A$ is trivial, i.e. $A$ is in $X$, or
  – $X$ is a superkey, i.e. $X \rightarrow R$ is in $F^+$

• Consider relation $R$ with FDs $F$. If $X \rightarrow A$ in $F$ over $R$ violates BCNF, i.e.,
  – $XA$ are all in $R$, and
  – $A$ is not in $X$, and
  – $X \rightarrow R$ is not in $F^+$
  → non-trivial FD
  → $X$ is not a superkey
Decomposition into BCNF

- Consider relation R with FDs F. If $X \rightarrow A$ in F over R violates BCNF, i.e.,
  - $XA$ are all in R, and
  - $A$ is not in $X$, and
  - $X \rightarrow R$ is not in $F^+$  
    → non-trivial FD  
    → $X$ is not a (super)key

- Then: decompose R into $R - A$ and $XA$.
- Repeated application of this idea will give us a collection of relations that are in BCNF; lossless join decomposition, and guaranteed to terminate.
BCNF Decomposition Example

- $R = (A, B, C)$
  $F = \{A \rightarrow B; B \rightarrow C\}$
  Key = \{A\}
- $R$ is not in BCNF ($B \rightarrow C$ but $B$ is not a superkey)
- Decomposition
  - $R_1 = (B, C)$
  - $R_2 = (A, B)$
BCNF Decomposition Example 2

• Assume relation schema CSJDPQV:
  \textit{Contracts} (\textit{contract\_id}, \textit{supplier}, \textit{project}, \textit{dept}, \textit{part}, \textit{qty}, \textit{value})
  – key C, JP → C, SD → P, J → S
• To deal with SD → P, decompose into SDP, CSJDQV.
• To deal with J → S, decompose CSJDQV into JS and CJDQV
• A tree representation of the decomposition:
BCNF Decomposition

• In general, several dependencies may cause violation of BCNF. The order in which we “deal with” them could lead to very different sets of relations!
How do we know R is in BCNF?

• If R has only two attributes, then it is in BCNF

• If F only uses attributes in R, then:
  – R is in BCNF if and only if for each X → Y in F (not $F^+$!), X is a superkey of R, i.e., X → R is in $F^+$ (not F!).
Checking for BCNF Violations

- List all non-trivial FDs
- Ensure that left hand side of each FD is a superkey
- We have to first find all the keys!
Checking for BCNF Violations

• Is Courses(course_num, dept_name, course_name, classroom, enrollment, student_name, address) in BCNF?

• FDs are:
  – course_num, dept_name → course_name
  – course_num, dept_name → classroom
  – course_num, dept_name → enrollment

• What is (course_num, dept_name)+?
  – {course_num, dept_name, course_name, classroom, enrollment}

• Therefore, the key is
  {course_num, dept_name, course_name, classroom, enrollment, student_name, address}

• The relation is not in BCNF
BCNF and Dependency Preservation

• In general, there may not be a dependency preserving decomposition into BCNF.
• Example: schema CSZ (city, street_name, zip_code) with FDs: CS \rightarrow Z, Z \rightarrow C
  
  (city, street_name) \rightarrow zip_code
  
  zip_code \rightarrow city

• Can’t decompose while preserving CS \rightarrow Z, but CSZ is not in BCNF.
Example Regarding Dependency Preservation

• $R = (A, B, C)$
  $F = \{A \rightarrow B, B \rightarrow C\}$
  – Can be decomposed in two different ways

• $R_1 = (A, B), \ R_2 = (B, C)$
  – Lossless-join decomposition:
    $R_1 \cap R_2 = \{B\}$ and $B \rightarrow BC$
  – Dependency preserving

• $R_1 = (A, B), \ R_2 = (A, C)$
  – Lossless-join decomposition:
    $R_1 \cap R_2 = \{A\}$ and $A \rightarrow AB$
  – Not dependency preserving
    (cannot check $B \rightarrow C$ without computing $R_1 \Join R_2$)
Dependency Preserving Decomposition

• Consider CSJDPQV, C is key, JP → C and SD → P.
  – BCNF decomposition: CSJDQV and SDP
  – Problem: Checking JP → C requires a join!

• 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))*
What FD on a decomposition?

- **Projection of set of FDs $F$:** If $R$ is decomposed into $X$, ... the 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 Decompositions (Contd.)

- Decomposition of R into X and Y is dependency preserving if \( (F_X \cup F_Y)^+ = F^+ \)
  - i.e., 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 \( F^+ \).

- Important to consider \( F^+ \), not \( F \), in this definition:
  - ABC, A → B, B → C, C → A, decomposed into AB and BC.
  - Is this dependency preserving? Is C → A preserved????

- Dependency preserving does not imply lossless join:
  - ABC, A → B, decomposed into AB and BC.

- And vice-versa!
Another example

- Assume CSJDPQV is decomposed into SDP, JS, CJDQV
  
  It is not dependency preserving w.r.t. the FDs: JP → C, SD → P and J → S.

- However, it is a lossless join decomposition.
- In this case, adding JPC to the collection of relations gives us a dependency preserving decomposition.
- JPC tuples stored only for checking FD!
Next: Third Normal Form

• There are some situations where
  – BCNF is not dependency preserving, and
  – efficient checking for FD violation on updates is important

• Solution: define a weaker normal form, called Third Normal Form (3NF)
  – Allows some redundancy (with resultant problems; we will see examples later)
  – But functional dependencies can be checked on individual relations without computing a join.
  – There is always a lossless-join, dependency-preserving decomposition into 3NF.
Summary of BCNF

• 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.
  – It is always possible to decompose a relation into a set of relations that are in BCNF such that:
    • the decomposition is lossless
    • it may not be possible to preserve dependencies.