Schema Refinement and Normalization

Nobody realizes that some people expend tremendous energy merely to be normal.

Albert Camus

**Normal Forms**
- Review FDs, Armstrong’s Axioms and Attr. Closures!
- Q1: is any refinement needed?!
- If a relation is in a normal form (BCNF, 3NF etc.):
  - we know that certain problems are avoided/minimized.
  - helps decide whether decomposing a relation is useful.
- Role of FDs in detecting redundancy:
  - Consider a relation R with 3 attributes, ABC.
  - No (non-trivial) FDs hold: There is no redundancy here.
  - Given A → B: If A is not a key, then several tuples could have the same A value, and if so, they’ll all have the same B value!
- 1st Normal Form — all attributes are atomic
  - i.e. the relational model
- 1st 2nd (of historical interest) 3rd Boyce-Codd…

**Boyce-Codd Normal Form (BCNF)**
- Reln R with FDs F is in BCNF if, for all X → A in F*
  - A ∈ X (called a trivial FD), or
  - X is a superkey for R.
- In other words: “R is in BCNF if the only non-trivial FDs over R are key constraints.”
- If R in BCNF, then every field of every tuple records information that cannot be inferred using FDs alone.
  - Say we know FD X → A holds this example relation:
    - Can you guess the value of the missing attribute?
    - Yes, so relation is not in BCNF

<table>
<thead>
<tr>
<th>X</th>
<th>Y</th>
<th>A</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>y1</td>
<td>7</td>
</tr>
<tr>
<td>5</td>
<td>y2</td>
<td>?</td>
</tr>
</tbody>
</table>

**Example (same as before)**

<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>Atishoo</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>

SNLRWH has FDs S → SNLRWH and R → W

Q: Is this relation in BCNF?
No, The second FD causes a violation; W values repeatedly associated with R values.

**Decomposition of a Relation Schema**
- If a relation is not in a desired normal form, it can be decomposed into multiple relations that each are in that normal form.
- Suppose that relation R contains attributes A1, ..., An. A 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
  - Every attribute of R appears as an attribute of at least one of the new relations.

**Decomposing a Relation**
- Easiest fix is to create a relation RW to store these associations, and to remove W from the main schema:

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Atishoo</td>
<td>48</td>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>30</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>40</td>
</tr>
</tbody>
</table>

Wages

<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>

Hourly_Emps

Hourly_Emps2

Q: Are both of these relations are now in BCNF?
• Decompositions should be used only when needed.
  – Q: potential problems of decomposition?
Problems with Decompositions

- There are three potential problems to consider:
  1) May be impossible to reconstruct the original relation! (Lossy Decomposition)
     - Fortunately, not in the SNLRWH example.
  2) Dependency checking may require joins (not Dependency Preserving)
     - Fortunately, not in the SNLRWH example.
  3) Some queries become more expensive.
     - e.g., How much does Guldu earn?

Tradeoff: Must consider these issues vs. redundancy. (Well, not usually #1)

Lossless Decomposition (example)

<table>
<thead>
<tr>
<th>S</th>
<th>N</th>
<th>L</th>
<th>R</th>
<th>H</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-22-3666</td>
<td>Atishoo</td>
<td>48</td>
<td>8</td>
<td>40</td>
</tr>
<tr>
<td>231-31-5368</td>
<td>Smiley</td>
<td>22</td>
<td>8</td>
<td>30</td>
</tr>
<tr>
<td>131-24-3650</td>
<td>Smethurst</td>
<td>35</td>
<td>5</td>
<td>30</td>
</tr>
<tr>
<td>434-26-3751</td>
<td>Guldu</td>
<td>35</td>
<td>5</td>
<td>32</td>
</tr>
<tr>
<td>612-67-4134</td>
<td>Madayan</td>
<td>35</td>
<td>8</td>
<td>40</td>
</tr>
</tbody>
</table>

Lossy Decomposition (example)

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

\[A \rightarrow B; \; C \rightarrow B\]

More on Lossless Decomposition

- The decomposition of R into X and Y is **lossless with respect to F** if and only if the closure of F contains:
  \[X \cap Y \rightarrow X, \text{ or } Y\]

  in example: decomposing ABC into AB and BC is lossy, because intersection (i.e., "B") is not a key of either resulting relation.

- Useful result: If \(W \rightarrow Z\) holds over R and \(W \cap Z\) is empty, then decomposition of R into R-Z and WZ is loss-less.

Lossless Join Decompositions

\[\pi_X(r) \rightarrow \pi_Y(r) = r\]

- It is always true that \(r \subseteq \pi_X(r) \triangleright \pi_Y(r)\)
  - In general, the other direction does not hold! If it does, the decomposition is lossless-join.

- 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 #1)

Lossless Decomposition (example)

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

\[A \rightarrow B; \; C \rightarrow B\]

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

But, now we can’t check \(A \rightarrow B\) without doing a join!
Dependency Preserving Decomposition

- Dependency preserving decomposition (Intuitive):
  - If R is decomposed into X, Y and Z, and we
    enforce the FDs that hold individually on X, on Y
    and on Z, then all FDs that were given to hold
    on R must also hold. (Avoids Problem #2 on
    our list.)
  - Why do we care??

- Projection of set of FDs F:
  - If R is decomposed into X and Y the projection of F
    on X (denoted F_X) is the set of FDs U → V in F^+ (closure of F, not just F)
    such that all of the attributes U, V are in X. (same holds
    for Y of course)

Decomposition into BCNF

- Consider relation R with FDs F. If X → Y violates
  BCNF, decompose R into R - Y and XY (guaranteed
to be loss-less).
  - Repeated application of this idea will give us a
    collection of relations that are in BCNF; lossless join decomposition,
    and guaranteed to terminate.
  - e.g., CSJDQV, key C, JP → C, SD → P, J → S
    - [contractid, supplierid, projectid, deptid, partid, qty, value]
    - To deal with SD → P, decompose into SDP, CSDQV.
    - To deal with J → S, decompose CSJDQV into JS and
      CJDQV
    - So we end up with: SDP, JS, and CJDQV
  - Note: several dependencies may cause violation of
    BCNF. The order in which we ‘deal with” them
    could lead to very different sets of relations!

Third Normal Form (3NF)

- Reln R with FDs F is in 3NF if, for all X → A in F^+
  A ⊆ X (called a trivial FD), or
  X is a superkey of R, or
  A is part of some candidate key (not superkey!) for R.
  (sometimes stated as "A is prime")

- Minimality of a key is crucial in third condition above!
  - If R is in BCNF, obviously in 3NF.
  - If R is in 3NF, some redundancy is possible. It is a
    compromise, used when BCNF not achievable (e.g., no
    “good” decom, or performance considerations).
    - Lossless-join, dependency-preserving decomposition of R
      into a collection of 3NF relations always possible.

What Does 3NF Achieve?

- If 3NF violated by X → A, one of the following holds:
  - X is a subset of some key K (‘partial dependency”)
    - We store (X, A) pairs redundantly.
    - e.g. Reserves SBDC (C is for credit card) with key SB and S → C
  - X is not a proper subset of any key. ("transitive dep.")
    - There is a chain of Fds X → Y →Z → A
    - So we can’t associate an X value with a K value unless we also associate an A
      value with an X value (different K’s, same X implies same A!)
  - problem with initial SNLRWH example.
  - But: even if R is in 3NF, these problems could arise.
    - e.g., Reserves SBDC (note: “C” is for credit card here), S → C, C → S is in 3NF (why?)
    - Even so, for each reservation of sailor S, same (S, C) pair is stored.
  - Thus, 3NF is indeed a compromise relative to BCNF.
    - You have to deal with the partial and transitive dependency issues
      in your application code!
An Aside: Second Normal Form

- Like 3NF, but allows transitive dependencies:
  - Reln R with FDs F is in 2NF if, for all \( X \rightarrow A \) in \( F^+ \)
    
    \( A \in X \) (called a trivial FD), or
    
    \( X \) is a superkey of R, or
    
    \( X \) is not part of any candidate key for R.
    (i.e. "\( X \) is not prime")
  - There’s no reason to use this in practice
    - And we won’t expect you to remember it

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) but does not ensure dependency preservation.
- To ensure dependency preservation, one idea:
  - If \( X \rightarrow Y \) is not preserved, add relation \( XY \).
  Problem is that \( XY \) may violate 3NF! e.g., consider the addition of CJP to "preserve’ JP \( \rightarrow C \). What if we also have \( J \rightarrow C \)?
  - Refinement: Instead of the given set of FDs \( F \), use a minimal cover for \( F \).

Minimal Cover for a Set of FDs

- **Minimal cover** \( G \) for a set of FDs \( F \):
  - Closure of \( F = \) 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:
  - \( A \rightarrow B, AC \rightarrow E, EF \rightarrow G \) and \( EF \rightarrow H \)
- M.C. implies Lossless-Join, Dep. Pres. Decomp!!!
  - (in book, p. 627)

Summary of Schema Refinement

- **BCNF**: each field contains information that cannot be inferred using only FDs.
  - ensuring BCNF is a good heuristic.
- **Not in BCNF?** Try decomposing into BCNF relations.
  - Must consider whether all FDs are preserved!
- **Lossless-join, dependency preserving decomposition into BCNF impossible?** Consider 3NF.
  - Same if BCNF decomp is unsuitable for typical queries
  - Decompositions should be carried out and/or re-examined while keeping performance requirements in mind.
- **Note:** even more restrictive Normal Forms exist (we don’t cover them in this course, but some are in the book.)