Normalization and FUNCTIONal Dependencies

Boo, Redundancy!

- **Redundancy**: root of several problems with relational schemas:
  - Redundant storage, insert/delete/update anomalies

- **Functional dependencies**: a form of integrity constraint that can identify schemas with such problems and suggest refinements.

- Main refinement technique: decomposition
  - Replacing ABCD with, say, AB and BCD, or ACD and ABD.

- Decomposition should be used judiciously:
  - Is there reason to decompose a relation?
  - What problems (if any) does the decomposition cause?
Hooray, FDs!

A functional dependency $X \rightarrow Y$ holds over relation schema $R$ if, for every allowable instance $r$ of $R$:

$t_1 \in r, \ t_2 \in r, \ \rho_X(t_1) = \rho_X(t_2)$

implies $\rho_Y(t_1) = \rho_Y(t_2)$

(where $t_1$ and $t_2$ are tuples; $X$ and $Y$ are sets of attributes)

In other words: $X \rightarrow Y$ means

Given any two tuples in $r$, if the $X$ values are the same, then the $Y$ values must also be the same. (but not vice versa)

Read "$\rightarrow$" as "determines"

Armstrong's Axioms ($X, Y, Z$ are sets of attributes):

- Reflexivity: If $X \subseteq Y$, then $X \rightarrow Y$
- Augmentation: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any $Z$
- Transitivity: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

These are sound and complete inference rules for FDs!

i.e., using AA you can compute all the FDs in $F^+$ and only these FDs.

Some additional rules (that follow from AA):

- Union: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
- Decomposition: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
**FD Inference**

- Given some FDs, we can usually infer additional FDs:
  \[ \text{title} \to \text{studio}, \text{star} \text{ implies } \text{title} \to \text{studio} \text{ and } \text{title} \to \text{star} \]
  \[ \text{title} \to \text{studio} \text{ and } \text{title} \to \text{star} \text{ implies } \text{title} \to \text{studio}, \text{star} \]
  \[ \text{title} \to \text{studio}, \text{studio} \to \text{star} \text{ implies } \text{title} \to \text{star} \]

  But,
  \[ \text{title, star} \to \text{studio} \text{ does NOT necessarily imply that } \text{title} \to \text{studio} \text{ or that } \text{star} \to \text{studio} \]

- An FD $f$ is **implied by** a set of FDs $F$ if $f$ holds whenever all FDs in $F$ hold.

- $F^+ = \text{closure of } F$ is the set of all FDs that are implied by $F$. (includes "trivial dependencies")

---

**Attribute Closure**

- Computing the closure of a set of FDs can be expensive. (Size of closure is exponential in # attrs!)

- Typically, we just want to check if a given FD $X \to Y$ is in the closure of a set of FDs $F$. An efficient check:
  - Compute **attribute closure** of $X$ (denoted $X^+$) wrt $F$.
    \[ X^+ = \text{Set of all attributes } A \text{ such that } X \to A \text{ is in } F^+ \]
    - $X^+ := X$
    - Repeat until no change: if there is an fd $U \to V$ in $F$ such that $U$ is in $X^+$, then add $V$ to $X^+$
  - Check if $Y$ is in $X^+$
  - Approach can also be used to find the keys of a relation.
    - If all attributes of $R$ are in the closure of $X$ then $X$ is a **superkey** for $R$. 
Anomalies (SNLRWH w00t)

<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>

• **Update anomaly**: Should we be allowed to modify W in only the 1st tuple of SNLRWH?

• **Insertion anomaly**: What if we want to insert an employee and don’t know the hourly wage for his or her rating? (or we get it wrong?)

• **Deletion anomaly**: If we delete all employees with rating 5, we lose the information about the wage for rating 5!

Boyce-Codd Normal Form (BCNF)

• Reln R with FDs F is in BCNF if, for all \( X \rightarrow A \) in \( F^+ \)
  - \( A \in X \) (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.
Decomposition

- Redundancy can be removed by "chopping" the relation into pieces (vertically!)
- FD's are used to drive this process.

R → W is causing the problems, so decompose SNLRWH into what relations?

<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>Attishoo</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>

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

Q: Are both of these relations are now in BCNF?
   • Indeed they are. But they're still just as fuzzy.

Consider relation R with FDs F. If \( X \rightarrow Y \) violates BCNF, decompose R into \( R - Y \) and \( XY \) (guaranteed to be lossless).

Repeating application of this idea will give us a collection of relations that are in BCNF; lossless join decomposition, and guaranteed to terminate.

E.g., CSJDPQV, key C, JP \( \rightarrow C \), SD \( \rightarrow P \), J \( \rightarrow S \)
   - \{contractid, supplierid, projectid, deptid, partid, qty, value\}

To deal with SD \( \rightarrow P \), decompose into SDP, CSJDQV.
To deal with J \( \rightarrow S \), decompose CSJDQV into JS and CJDQV.

So we end up with: SDP, JS, and CJDQV.

Note: several dependencies may cause violation of BCNF.
Problems with Decomposition

- **There are three potential problems to consider:**
  1) May be impossible to reconstruct the original relation! (Lossy Decomposition)
  2) Dependency checking may require joins (not Dependency Preserving)
  3) Some queries become more expensive.

It is essential that all decompositions used to deal with redundancy be lossless! (Avoids Problem #1)

## Lossless Join Decomposition

<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>Attishoo</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>

<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></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

\[ S = \begin{array}{cccc}
123-22-3666 & Attishoo & 48 & 8 & 10 & 40 \\
231-31-5368 & Smiley & 22 & 8 & 10 & 30 \\
131-24-3650 & Smethurst & 35 & 5 & 7 & 30 \\
434-26-3751 & Guldu & 35 & 5 & 7 & 32 \\
612-67-4134 & Madayan & 35 & 8 & 10 & 40 \\
\end{array} \]
Lossy Decomposition

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

  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.
Dep. Preserving Composition

• 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 \( U \rightarrow V \) 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)

Third Normal Form (3NF)

• Reln R with FDs F is in 3NF if, for all \( X \rightarrow A \) in \( F^+ \)
  \( A \in X \) (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” decomp, or performance considerations).
  – Lossless-join, dependency-preserving decomposition of R into a collection of 3NF relations always possible.
Minimal Cover

- **Minimal cover** $G$ 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 $EF \rightarrow H$
- M.C. implies Lossless-Join, Dep. Pres. Decomp!!!
  - (in book, p. 627)

Summary

- 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.)