Satisfiability: Does an FD hold?

- Satisfiability of FDs
- Given: FD $X \rightarrow Y$ and relation $R$
- Output: Does $R$ satisfy $X \rightarrow Y$?
- Algorithm:
  - a. Sort $R$ on $X$
  - b. Do all the tuples with equal $X$ values agree on their $Y$ values?

Original Database

<table>
<thead>
<tr>
<th>St-Name</th>
<th>Status</th>
<th>Course-Name</th>
<th>Course-#</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ben</td>
<td>senior</td>
<td>db</td>
<td>670</td>
<td>500</td>
</tr>
<tr>
<td>Ben</td>
<td>senior</td>
<td>ds</td>
<td>680</td>
<td>500</td>
</tr>
<tr>
<td>Dan</td>
<td>freshman</td>
<td>db</td>
<td>670</td>
<td>400</td>
</tr>
<tr>
<td>Dan</td>
<td>junior</td>
<td>ds</td>
<td>680</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

Checking for above functional dependencies

(St-Name, Status) → (Salary)? YES

<table>
<thead>
<tr>
<th>St-Name</th>
<th>Status</th>
<th>Salary</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ben</td>
<td>senior</td>
<td>500</td>
</tr>
<tr>
<td>Ben</td>
<td>senior</td>
<td>500</td>
</tr>
</tbody>
</table>

(St-Name, Status) → (Course-#)? NO

<table>
<thead>
<tr>
<th>St-Name</th>
<th>Status</th>
<th>Course-#</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ben</td>
<td>senior</td>
<td>670</td>
</tr>
<tr>
<td>Dan</td>
<td>freshman</td>
<td>670</td>
</tr>
<tr>
<td>Dan</td>
<td>junior</td>
<td>680</td>
</tr>
<tr>
<td>Dan</td>
<td>senior</td>
<td>680</td>
</tr>
</tbody>
</table>
Inference of FD’s

• The set of all FDs on R-a brute force algorithm:
  – Find all possible candidate FDs on attributes of R
  – For each candidate apply the satisfiability algorithm
  – Time consuming!
• Alternative: Inference
  – Some FDs may be inferred from other FDs
  – given: {St-Name, Course-#} → Salary
  – implies: {St-Name, Course-, Status} → Salary

• Basically
  • Start from some set F of FDs
  • Derive all possible FDs from F

Based on Rules of Inference

• The set of all FDs derivable from F is called the closure F⁺ of F

<p>| | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
</tr>
</thead>
</table>
| F | a (St-Name) → Status  
|   | b. (St-Name, Course-) → Salary  
| F⁺ | 1. (St-Name) → St-Name  
|   | 2. (Course-, Status) → (Course-, Status)  
|   | 3. (Course-, Status) → Status  
|   |   
|   | i. From (a) (St-Name, Course-) → (Status, Course-)  
|   |   |
|   |   |

Armstrong’s Rules of Inference

• Reflexive (Intuition: superset implies subset (need not be proper))
  – (Name, Sex) → Name  
  – (Name, Sex) → (Name, Sex)
• Transitivity
  – X→Y, Y→Z implies X→Z
• Augmentation (augment both sides of the implication)
  – (Name, Sex) → Name implies  
  – (Name, Sex, Age) → (Name, Age)
• Rules are sound and complete
  – Produces only FD’s in the closure  
  – Produces all FD’s in closure
Augmenting Armstrong’s rules

- Decomposition
  - Numb \( \rightarrow \) \{Name,Age\} implies Numb \( \rightarrow \) Name
- Union
  - \( X \rightarrow Y \) and \( X \rightarrow Z \) implies \( X \rightarrow YZ \)
- *The above augment basic rules but do not add to power*

---

**Given FDs:**

1. \( AB \rightarrow E \)
2. \( BE \rightarrow I \)
3. \( E \rightarrow C \)
4. \( CI \rightarrow D \)

**Find a derivation for \( AB \rightarrow CD \):**

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>( AB \rightarrow E )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>b</td>
<td>( AB \rightarrow AB )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>c</td>
<td>( AB \rightarrow B )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>d</td>
<td>( AB \rightarrow BB )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>e</td>
<td>( EE \rightarrow I )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>f</td>
<td>( AB \rightarrow I )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>g</td>
<td>( E \rightarrow C )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>h</td>
<td>( AB \rightarrow C )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>i</td>
<td>( AB \rightarrow CI )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>j</td>
<td>( CI \rightarrow D )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>k</td>
<td>( AB \rightarrow D )</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>l</td>
<td>( AB \rightarrow CD )</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

---

**Dis-Prove by Counter Example**

**Given FDs:**

1. \( AB \rightarrow E \)
2. \( BE \rightarrow I \)
3. \( E \rightarrow C \)
4. \( CI \rightarrow D \)

**Question** Is \( B \rightarrow CD \) in the closure?

No. The above FDs, but not \( B \rightarrow CD \), are satisfied by

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
<th>I</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>e1</td>
<td>d1</td>
<td>e1</td>
<td>d1</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>e2</td>
<td>d2</td>
<td>e2</td>
<td>d2</td>
<td></td>
</tr>
</tbody>
</table>

**Tuples:**

- Tuple 1
- Tuple 2
Algorithm: Computing FD closures

- The closure $F^+$ of $F$ is the set of FD’s inferred from $F$.
- Algorithm:
  - $F^+ := F$
  - repeat
    - apply Armstrong’s inference rules on $F^+$
    - $F^+ := F$
  - Until $F^+$ is not augmented
  - end

Determining closure of X (attributes) under F (set of FD’s)

- Given $F$:
  - 1. $AB \Rightarrow E$
  - 2. $BE \Rightarrow I$
  - 3. $E \Rightarrow C$
  - 4. $CI \Rightarrow D$
- Derive $\{A, B\}^+$:
  a. $\{A, B, E\}$ (given $X$)
  b. $\{A, B, E, I\}$ (FD #2 in $F$)
  c. $\{A, B, E, I, C\}$ (FD #3 in $F$)
  d. $\{A, B, E, I, C, D\}$ (FD #4 in $F$)

Requirements on Decompositions

- **Lossless Decomposition**
- Sometimes not possible
  - $A \subseteq B \subseteq C \subseteq D \subseteq E$
  - Given the following fd’s
    - $A \Rightarrow BCDE$
    - $CD \Rightarrow E$
    - $CE \Rightarrow B$
  - 3NF would dictate
    - Either $\{A, B, C, D\}$ & $\{C, D, E\}$
    - Or $\{A, B, C, E\}$ & $\{C, E, B\}$
  - In either case you lose a functional dependency
Dependency Preserving 3NF

- Objective: Introduce a dependency-preserving decomposition algorithm 3NF
- The subset $F_X$ of the closure $F^+$ which uses only attributes of $X$ is the projection of $F$ on $X$
- A decomposition of $R$ into $R(X)$ and $R(Y)$ is dependency preserving if $F^+ = (F_X \cup F_Y)^+$
- Trick is to use the minimal cover of $F$ to drive decomposition

Minimal Cover

- $F$ is a minimal set of FDs if each $X \rightarrow Y$ is
  - $F = \{A \rightarrow BE, AB \rightarrow DE, AC \rightarrow G\}$
  - Canonical: $|Y| = 1$ (use decomposition)
    - $A \rightarrow B, A \rightarrow E, AB \rightarrow D, AB \rightarrow E, AC \rightarrow G$
  - Left-reduced: $X$ can’t be replaced by a subset
    - $A \rightarrow B A \rightarrow E, AC \rightarrow G$
    - $A \rightarrow D$ (How?)
  - Non-redundant: $X \rightarrow Y$ can’t be removed

Dependency-Preserving 3NF Decomposition Algorithm

- Find minimal cover
- Put FDs agreeing on the left-hand-side in the same schema
- Have extra schema for unaccounted attributes
- Example
  - $R = \{A, B, C, D, E, G, I, J\}$
  - $A \rightarrow B$
  - $A \rightarrow E$
  - $A \rightarrow D$
  - $AC \rightarrow G$
  - Resulting Schemas
    - $ABDE$ $ACG$ $IJ$

Note this is not the decomposition you would get using 3NF normalization, but this is in 3NF
Lossless Joins

- A Decomposition is a set of projections of relational schemas
- A natural join should return the original table

\[
\begin{array}{c|c|c}
R & A & B & C \\
\hline
a1 & b1 & c1 \\
a2 & b2 & c2 \\
a3 & b1 & c3 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\pi_{A\delta}(R) & \pi_{B\gamma}(R) & A & B & C \\
\hline
a1 & b1 & c1 \\
a2 & b2 & c2 \\
a3 & b1 & c3 \\
\end{array}
\]

Not Lossless!

\[
\pi_{A\delta}(R) \ast \pi_{B\gamma}(R)
\]

Lossless decomposition

\[
\begin{array}{c|c|c}
R & A & B & C \\
\hline
a1 & b1 & c1 \\
a2 & b2 & c2 \\
a3 & b1 & c3 \\
\end{array}
\]

\[
\begin{array}{c|c|c}
\pi_{A\delta}(R) \cap \pi_{B\gamma}(R) = \emptyset \\
\end{array}
\]

R1, R2 is a lossless join decomposition of R iff the attributes common to R1 and R2 contain a key for at least one of the involved relations.
### Dependency-Preserving Losses-Join 3NF Decomposition Algorithm

- Find minimal cover
- Put FDs agreeing on the left-hand-side in the same schema
- Have extra schema for a key, if none of the above schemas contain a key
- \( R = \{ A, B, C, D, E, G, I, J \} \)
- \( F = \{ A \rightarrow B, A \rightarrow E, A \rightarrow D, AC \rightarrow G \} \)
- Resulting Schemas:
  - A B E D
  - AC G
  - A C I J

**NOTE THIS IS THE DECOMPOSITION YOU WOULD GET USING 3NF NORMALIZATION.**

### Facts

- Schemas can always employ lossless-join dependency-preserving decompositions to achieve 3NF
- Determining whether a relationship schema satisfies 3NF is NP-complete. Hence, automated normalization is tough.
- Not all violations to BCNF can be resolved through dependency-preserving decompositions
  - MANAGER \( \rightarrow \) PROJECT DEPT
  - \( \{ \text{PROJECT, DEPT} \} \rightarrow \) MANAGER
  - MANAGER \( \rightarrow \) DEPT
- Doesn’t satisfy BCNF but is in 3NF
- BCNF decompositions can’t always preserve dependencies