Inference Rules for FD’s
(continued)

Transitive Closure Rule

If

\[ A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m \]

and

\[ B_1, B_2, \ldots, B_m \rightarrow C_1, C_2, \ldots, C_p \]

then

\[ A_1, A_2, \ldots, A_n \rightarrow C_1, C_2, \ldots, C_p \]

Why?
Closure of a set of FDs

• It is not sufficient to consider just the given set of FDs
• We need to consider all FDs that hold
• Given F, more FDs can be inferred
• Such FDs are said to be logically implied by F
• \( F^+ \) is the set of all FDs logically implied by F
• We can compute \( F^+ \) using formal definition of FD
• If F were large, this process would be lengthy & cumbersome
• Axioms or Rules of Inference provide simpler technique
• Armstrong’s Axioms
Inference Rules for FDs

Armstrong's inference rules:

IR1. (Reflexive) If \( Y \subseteq X \), then \( X \Rightarrow Y \)

IR2. (Augmentation) If \( X \Rightarrow Y \), then \( XZ \Rightarrow YZ \)

(Note: \( XZ \) stands for \( X \cup Z \))

IR3. (Transitive) If \( X \Rightarrow Y \) and \( Y \Rightarrow Z \), then \( X \Rightarrow Z \)

IR1, IR2, IR3 form a sound & complete set of inference rules

Never generates any wrong FD
Generate all FDs that hold
Example

• \( R = (A, B, C, G, H, I) \)
• \( F = \{ A \rightarrow B, \ A \rightarrow C, \ CG \rightarrow H, \ CG \rightarrow I, \ B \rightarrow H \} \)

• some members of \( F^+ \)
  – \( A \rightarrow H \)
    • by transitivity from \( A \rightarrow B \) and \( B \rightarrow H \)
  – \( AG \rightarrow I \)
    • by augmenting \( A \rightarrow C \) with \( G \), to get \( AG \rightarrow CG \) and then transitivity with \( CG \rightarrow I \)
  – \( CG \rightarrow HI \)
    • By union rule
• Based on FDs that take into account all candidate keys of a relation

• For a relation with only 1 CK, 3NF & BCNF are equivalent

• A relation is said to be in BCNF if every determinant is a CK

• Is PLOTS in BCNF?
• NO
For every functional dependency $X \rightarrow Y$ in a set $F$ of functional dependencies over relation $R$, either:

- $Y$ is a subset of $X$, or,
- $X$ is a superkey of $R$, or
- $Y$ is a subset of $K$ for some key $K$ of $R$.

### 3NF Schema

<table>
<thead>
<tr>
<th>Account</th>
<th>Client</th>
<th>Office</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Joe</td>
<td>1</td>
</tr>
<tr>
<td>B</td>
<td>Mary</td>
<td>1</td>
</tr>
<tr>
<td>A</td>
<td>John</td>
<td>1</td>
</tr>
<tr>
<td>C</td>
<td>Joe</td>
<td>2</td>
</tr>
</tbody>
</table>

Client, Office $\rightarrow$ Client, Office, Account
Account $\rightarrow$ Office
Goals of Decomposition

1. Lossless Joins
   Want to be able to reconstruct big (e.g. universal) relation by joining smaller ones (using natural joins)
   (i.e. $R_1 \bowtie R_2 = R$)

2. Dependency preservation
   Want to minimize the cost of global integrity constraints based on FD’s
   (i.e. avoid big joins in assertions)

3. Redundancy Avoidance
   Avoid unnecessary data duplication (the motivation for decomposition)

Why important?
   LJ: information loss
   DP: efficiency (time)
   RA: efficiency (space), update anomalies
Decomposition Goal #2: Dependency preservation

Example: Given $F = \{ A \rightarrow B, AB \rightarrow D, C \rightarrow D \}$

consider $R = R_1 \cup R_2$ s.t.
$R_1 = (A, B, D), \quad R_2 = (C, D)$

(1) $F^+ = \{ A \rightarrow BD, C \rightarrow D \}^+$
(2) $G = \{ A \rightarrow BD, C \rightarrow D, ... \}^+$

(3) $F^+ = G^+$
   note: $G^+$ cannot introduce new FDs not in $F^+$

Decomposition is DP
Example

• $R = (A, B, C)$
  $F = \{ A \rightarrow B, \quad B \rightarrow C \}$
  Key = \{A\}

• $R$ is not in BCNF ($B \rightarrow C$ but $B$ is not superkey)

• Decomposition $R_1 = (A, B), \quad R_2 = (B, C)$
  – $R_1$ and $R_2$ in BCNF
  – Lossless-join decomposition
  – Dependency preserving
3NF Decomposition Algorithm

Let $F_c$ be a canonical cover for $F$;

\[ i := 0; \]

for each functional dependency $\alpha \rightarrow \beta$ in $F_c$ do

if none of the schemas $R_j$, $1 \leq j \leq i$ contains $\alpha \beta$ then begin

\[ i := i + 1; \]

\[ R_i := \alpha \beta \]

end

if none of the schemas $R_j$, $1 \leq j \leq i$ contains a candidate key for $R$ then begin

\[ i := i + 1; \]

\[ R_i := \text{any candidate key for } R; \]

end

/* Optionally, remove redundant relations */

repeat

if any schema $R_j$ is contained in another schema $R_k$ then /* delete $R_j$ */

\[ R_j = R;; \]

\[ i := i - 1; \]

return $(R_1, R_2, ..., R_i)$
BCNF Decomposition Algorithm

\[
\text{result} := \{ R \};
\]
\[
\text{done} := \text{false};
\]
\[
\text{compute } F^+;
\]
\[
\text{while (not done) do}
\]
\[
\quad \text{if (there is a schema } R_i \text{ in result that is not in BCNF)}
\]
\[
\quad \quad \text{then begin}
\]
\[
\quad \quad \quad \text{let } \alpha \rightarrow \beta \text{ be a nontrivial functional dependency that holds on } R_i \text{ such that } \alpha \rightarrow R_i \text{ is not in } F^+,
\]
\[
\quad \quad \quad \quad \text{and } \alpha \cap \beta = \emptyset;
\]
\[
\quad \quad \quad \quad \text{result} := (\text{result} - R_i) \cup (R_i - \beta) \cup (\alpha, \beta);
\]
\[
\quad \quad \text{end}
\]
\[
\quad \text{else done := true;}
\]

Note: each \( R_i \) is in BCNF, and decomposition is lossless-join.