Chapter 11, Relational Database Design
Algorithms and Further Dependencies

- Normal forms are insufficient on their own as a criteria for a good relational database schema design.

- The relations in a database must collectively satisfy two other properties - dependency preservation property and lossless (or nonadditive) join property - to qualify as a good design.

11.1 Properties of Relational Decompositions

11.1.1 Relation Decomposition and Insufficiency of Normal Forms

- The relational database design algorithms discussed in this chapter start from a single universal relation schema $R = \{A_1, A_2, \ldots, A_n\}$ that includes all the attributes of the database. Using the functional dependencies $F$, the algorithms decompose $R$ into a set of relation schemas $DECOMP = \{R_1, R_2, \ldots, R_m\}$ that will become the relational database schema.

- Examine an individual relation $R_i$ to test whether it is in a higher normal form does not guarantee a good design (decomposition); rather, a set of relations that together form the relation database schema must possess certain additional properties to ensure a good design.

  - **Attribute preservation** property: Each attribute in $R$ will appear in at least one relation $R_i$ in the decomposition so that no attributes are 'lost'; formally we have
    $$\bigcup_{i=1}^{m} R_i = R$$

  - **Dependency preservation** property: See Figure 10.12 (Fig 14.12 on e3). Discuss on 11.1.2

  - **Lossless (nonadditive) join** property: If we decompose EMP_PROJ in Figure 10.2 (Fig 14.2 on e3) to EMP_LOCS and EMP_PROJ1 in Figure 10.5 (Fig 14.5 on e3), it will violate the nonadditive join property. Discuss on 11.1.3
11.1.2 Decomposition and Dependency Preservation

- It is not necessary that the exact dependencies specified in $F$ on $R$ appear themselves in individual relations of the composition $DECOMP$. It is sufficient that the union of the dependencies that hold on individual relations in $DECOMP$ be equivalent to $F$.

- The projection of $F$ on $R_i$, denoted by $\pi_{R_i}(F)$, is the set of dependencies $X \rightarrow Y$ in $F^+$ such that the attributes in $X \cup Y$ are contained in $R_i$.

- A decomposition $DECOMP = \{R_1, R_2, \ldots, R_m\}$ of $R$ is dependency-preserving with respect to $F$ if

$$((\pi_{R_1}(F)) \cup \ldots \cup (\pi_{R_m}(F)))^+ = F^+$$

- **Claim 1:** It is always possible to find a dependency-preserving decomposition $DECOMP$ with respect to $F$ such that each relation $R_i$ in $DECOMP$ is in 3NF.

11.1.3 Decomposition and Lossless (Nonadditive) Joins

- A decomposition $DECOMP = \{R_1, R_2, \ldots, R_m\}$ of $R$ has the lossless join property with respect to the set of dependencies $F$ on $R$ if, for every relation state $r$ of $R$ that satisfies $F$, the following holds, where $*$ is the NATUAL JOIN of all the relations in $DECOMP$:

$$*(\pi_{R_1}(r), \ldots, \pi_{R_m}(r)) = r$$

- The decomposition of EMP_PROJ(SSN, PNUMBER, HOURS, ENAME, PNAME, PLOCATION) from Figure 10.3 (Fig 14.3 on e3) into EMP_LOCS(ENAME, PLOCATION) and EMP_PROJ1(SSN, PNUMBER, HOURS, PNAME, PLOCATION) in Figure 10.5 (Fig 14.5 on e3) does not have the lossless join property.

- Another example: decompose $R$ to $R_1$ and $R_2$ as follows.

<table>
<thead>
<tr>
<th>R</th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$R_1$</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>1</td>
<td>4</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>$R_2$</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>3</td>
<td>4</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>3</td>
<td>5</td>
<td></td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>2</td>
<td></td>
</tr>
</tbody>
</table>
Is \( R \) equal to \( R_1 \ast R_2 \)?

| \( R_1 \ast R_2 \) |
|---|---|---|---|
| A | B | C | D |
| 1 | 2 | 3 | 4 |
| 1 | 2 | 3 | 5 |
| 4 | 2 | 3 | 4 |
| 4 | 2 | 3 | 5 |
| 3 | 1 | 4 | 2 |

- **Algorithm 11.1** Testing for the lossless (nonadditive) join property

**Input:** A universal relation \( R \), a decomposition \( DECOMP = \{R_1, R_2, \ldots, R_m\} \) of \( R \), and a set \( F \) of functional dependencies.

- 1. Create an initial matrix \( S \) with one row \( i \) for each relation \( R_i \) in \( DECOMP \), and one column \( j \) for each attribute \( A_j \) in \( R \).
- 2. Set \( S(i, j) := b_{ij} \) for all matrix entries.
- 3. For each row \( i \)
  - For each column \( j \)
    - If \( R_i \) includes attribute \( A_j \)
      - Then set \( S(i, j) := a_j \)
- 4. Repeat the following loop until a complete loop execution results in no changes to \( S \)
  - For each \( X \rightarrow Y \) in \( F \)
    - For all rows in \( S \) which has the same symbols in the columns corresponding to attributes in \( X \)
      - make the symbols in each column that correspond to an attribute in \( Y \) be the same in all these rows as follows:
        - if any of the rows has an “a” symbol for the column, set the other rows to the same “a” symbol in the column.
        - If no “a” symbol exists for the attribute in any of the rows, choose one of the “b” symbols that appear in one of the rows for the attribute and set the other rows to that same “b” symbol in the column.
5. If a row is made up entirely of “a” symbols, then the decomposition has the
lossless join property; otherwise it does not.

- Example:

\[ R = \{ \text{SSN, ENAME, PNUMBER, PNAME, PLOCATION, HOURS} \} \]
\[ F = \{ \text{SSN} \rightarrow \text{ENAME}, \text{PNUMBER} \rightarrow \{ \text{PNAME, PLOCATION} \}, \{ \text{SSN}, \text{PNUMBER} \} \rightarrow \text{HOURS} \} \]
\[ \text{DECOMP} = \{ R_1, R_2, R_3 \} \]
\[ R_1 = \{ \text{SSN, ENAME} \} \]
\[ R_2 = \{ \text{PNUMBER, PNAME, PLOCATION} \} \]
\[ R_3 = \{ \text{SSN, PNUMBER, HOURS} \} \]

<table>
<thead>
<tr>
<th></th>
<th>SSN</th>
<th>ENAME</th>
<th>PNUMBER</th>
<th>PNAME</th>
<th>PLOCATION</th>
<th>HOURS</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>( a_1 )</td>
<td>( a_2 )</td>
<td>( b_{13} )</td>
<td>( b_{14} )</td>
<td>( b_{15} )</td>
<td>( b_{16} )</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>( b_{21} )</td>
<td>( b_{22} )</td>
<td>( a_3 )</td>
<td>( a_4 )</td>
<td>( a_5 )</td>
<td>( b_{26} )</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>( a_1 )</td>
<td>( b_{32} )</td>
<td>( a_3 )</td>
<td>( b_{34} )</td>
<td>( b_{35} )</td>
<td>( a_6 )</td>
</tr>
</tbody>
</table>

- Try \( \text{SSN} \rightarrow \text{ENAME} \):
  \[ a_1 \rightarrow b_{32} a_2 \]

- Try \( \text{PNUMBER} \rightarrow \{ \text{PNAME, PLOCATION} \} \):
  \[ a_3 \rightarrow b_{34} a_4 \]
  \[ a_3 \rightarrow b_{35} a_5 \]

In row 3, all symbols are \( a \). The decomposition \( \text{DECOMP} \) has the lossless join property.

- Another example:

\[ R = \{ A, B, C, D, E \} \]
\[ F = \{ A \rightarrow C, B \rightarrow C, C \rightarrow D, DE \rightarrow C, CE \rightarrow A \} \]
\[ \text{DECOMP} = \{ R_1, R_2, R_3, R_4, R_5 \} \]
\[ R_1 = \{A, D\} \]
\[ R_2 = \{A, B\} \]
\[ R_3 = \{B, E\} \]
\[ R_4 = \{C, D, E\} \]
\[ R_5 = \{A, E\} \]

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>( a_1 )</td>
<td>( b_{12} )</td>
<td>( b_{13} )</td>
<td>( a_4 )</td>
<td>( b_{15} )</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>( a_1 )</td>
<td>( a_2 )</td>
<td>( b_{23} )</td>
<td>( b_{24} )</td>
<td>( b_{25} )</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>( b_{31} )</td>
<td>( a_2 )</td>
<td>( b_{33} )</td>
<td>( b_{34} )</td>
<td>( a_5 )</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>( b_{41} )</td>
<td>( b_{42} )</td>
<td>( a_3 )</td>
<td>( a_4 )</td>
<td>( a_5 )</td>
</tr>
<tr>
<td>( R_5 )</td>
<td>( a_1 )</td>
<td>( b_{52} )</td>
<td>( b_{53} )</td>
<td>( b_{54} )</td>
<td>( a_5 )</td>
</tr>
</tbody>
</table>

- For each FDs in \( F \) (first loop):
  
  * Try \( A \rightarrow C \):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>( a_1 )</td>
<td>( b_{12} )</td>
<td>( b_{13} )</td>
<td>( a_4 )</td>
<td>( b_{15} )</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>( a_1 )</td>
<td>( a_2 )</td>
<td>( b_{23} )</td>
<td>( b_{24} )</td>
<td>( b_{25} )</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>( b_{31} )</td>
<td>( a_2 )</td>
<td>( b_{33} )</td>
<td>( b_{34} )</td>
<td>( a_5 )</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>( b_{41} )</td>
<td>( b_{42} )</td>
<td>( a_3 )</td>
<td>( a_4 )</td>
<td>( a_5 )</td>
</tr>
<tr>
<td>( R_5 )</td>
<td>( a_1 )</td>
<td>( b_{52} )</td>
<td>( b_{53} )</td>
<td>( b_{54} )</td>
<td>( a_5 )</td>
</tr>
</tbody>
</table>

* Try \( B \rightarrow C \):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>( R_1 )</td>
<td>( a_1 )</td>
<td>( b_{12} )</td>
<td>( b_{13} )</td>
<td>( a_4 )</td>
<td>( b_{15} )</td>
</tr>
<tr>
<td>( R_2 )</td>
<td>( a_1 )</td>
<td>( a_2 )</td>
<td>( b_{13} )</td>
<td>( b_{24} )</td>
<td>( b_{25} )</td>
</tr>
<tr>
<td>( R_3 )</td>
<td>( b_{31} )</td>
<td>( a_2 )</td>
<td>( b_{13} )</td>
<td>( b_{34} )</td>
<td>( a_5 )</td>
</tr>
<tr>
<td>( R_4 )</td>
<td>( b_{41} )</td>
<td>( b_{42} )</td>
<td>( a_3 )</td>
<td>( a_4 )</td>
<td>( a_5 )</td>
</tr>
<tr>
<td>( R_5 )</td>
<td>( a_1 )</td>
<td>( b_{52} )</td>
<td>( b_{13} )</td>
<td>( b_{54} )</td>
<td>( a_5 )</td>
</tr>
</tbody>
</table>
* Try $C \rightarrow D$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$b_{12}$</td>
<td>$b_{13}$</td>
<td>$a_4$</td>
<td>$b_{15}$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$b_{13}$</td>
<td>$b_{14}$</td>
<td>$a_4$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>$b_{31}$</td>
<td>$a_2$</td>
<td>$b_{13}$</td>
<td>$b_{34}$</td>
<td>$a_4$</td>
</tr>
<tr>
<td>$R_4$</td>
<td>$b_{41}$</td>
<td>$b_{42}$</td>
<td>$a_3$</td>
<td>$a_4$</td>
<td>$a_5$</td>
</tr>
<tr>
<td>$R_5$</td>
<td>$a_1$</td>
<td>$b_{52}$</td>
<td>$b_{13}$</td>
<td>$b_{54}$</td>
<td>$a_4$</td>
</tr>
</tbody>
</table>

* Try $DE \rightarrow C$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$b_{12}$</td>
<td>$b_{13}$</td>
<td>$a_4$</td>
<td>$b_{15}$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$b_{13}$</td>
<td>$b_{14}$</td>
<td>$a_4$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>$b_{31}$</td>
<td>$a_2$</td>
<td>$b_{13}$</td>
<td>$a_3$</td>
<td>$a_4$</td>
</tr>
<tr>
<td>$R_4$</td>
<td>$b_{41}$</td>
<td>$b_{42}$</td>
<td>$a_3$</td>
<td>$a_4$</td>
<td>$a_5$</td>
</tr>
<tr>
<td>$R_5$</td>
<td>$a_1$</td>
<td>$b_{52}$</td>
<td>$b_{13}$</td>
<td>$a_3$</td>
<td>$a_4$</td>
</tr>
</tbody>
</table>

* Try $CE \rightarrow A$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
<th>E</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$b_{12}$</td>
<td>$b_{13}$</td>
<td>$a_4$</td>
<td>$b_{15}$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$b_{13}$</td>
<td>$b_{14}$</td>
<td>$a_4$</td>
</tr>
<tr>
<td>$R_3$</td>
<td>$b_{31}$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$a_3$</td>
<td>$a_4$</td>
</tr>
<tr>
<td>$R_4$</td>
<td>$b_{41}$</td>
<td>$a_1$</td>
<td>$b_{42}$</td>
<td>$a_3$</td>
<td>$a_4$</td>
</tr>
<tr>
<td>$R_5$</td>
<td>$a_1$</td>
<td>$b_{52}$</td>
<td>$a_3$</td>
<td>$a_4$</td>
<td>$a_5$</td>
</tr>
</tbody>
</table>

- The third row is made up entirely of $a_i$ symbols. The decomposition $DECOMP$ has the lossless join property.

- Another example:
  
  $R = \{A, B, C\}$
  
  $F = \{AB \rightarrow C, C \rightarrow B\}$
  
  $DECOMP = \{R_1, R_2\}$

- If $R_1 = \{A, B\}$ and $R_2 = \{B, C\}$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$a_2$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$b_{21}$</td>
<td>$a_2$</td>
</tr>
</tbody>
</table>
* For each FDs in $F$ (first loop):

- Try $AB \rightarrow C$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$b_{13}$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$b_{21}$</td>
<td>$a_2$</td>
<td>$a_3$</td>
</tr>
</tbody>
</table>

- Try $C \rightarrow B$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$b_{13}$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$b_{21}$</td>
<td>$a_2$</td>
<td>$a_3$</td>
</tr>
</tbody>
</table>

- Complete loop execution results in no changes to the matrix and the matrix does not contain a row with all $a_i$ symbols. This decomposition does not have lossless join property.

- If $R_1 = \{A, C\}$ and $R_2 = \{B, C\}$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$b_{12}$</td>
<td>$a_3$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$b_{21}$</td>
<td>$a_2$</td>
<td>$a_3$</td>
</tr>
</tbody>
</table>

* For each FDs in $F$ (first loop):

- Try $AB \rightarrow C$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$b_{12}$</td>
<td>$a_3$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$b_{21}$</td>
<td>$a_2$</td>
<td>$a_3$</td>
</tr>
</tbody>
</table>

- Try $C \rightarrow B$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$b_{12}$</td>
<td>$a_3$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$b_{21}$</td>
<td>$a_2$</td>
<td>$a_3$</td>
</tr>
</tbody>
</table>

- Row 1 contains all $a_i$ symbols. This decomposition has the lossless join property.
− If $R_1 = \{A, B\}$ and $R_2 = \{A, C\}$

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$b_{13}$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$a_1$</td>
<td>$b_{22}$</td>
<td>$a_3$</td>
</tr>
</tbody>
</table>

* For each $FD$s in $F$ (first loop):
  · Try $AB \rightarrow C$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$b_{13}$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$a_1$</td>
<td>$b_{22}$</td>
<td>$a_3$</td>
</tr>
</tbody>
</table>

  · Try $C \rightarrow B$:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>$R_1$</td>
<td>$a_1$</td>
<td>$a_2$</td>
<td>$b_{13}$</td>
</tr>
<tr>
<td>$R_2$</td>
<td>$a_1$</td>
<td>$b_{22}$</td>
<td>$a_3$</td>
</tr>
</tbody>
</table>

  · Complete loop execution results in no changes to the matrix and the matrix does not contain a row with all $a_i$ symbols. This decomposition does not have lossless join property.

11.1.4 Testing Binary Decompositions for the Nonadditive Join Property

• Property LJ1 below is a handy way to decompose a relation into two relations.

  − **Property LJ1** A decomposition $DECOMP = \{R_1, R_2\}$ of $R$ has the lossless join property with respect to a set of functional dependencies $F$ on $R$ *if and only if* either
    * The FD $((R_1 \cap R_2) \rightarrow (R_1 - R_2))$ is in $F^+$, or
    * The FD $((R_1 \cap R_2) \rightarrow (R_2 - R_1))$ is in $F^+$. 

11.1.5 Successive Lossless (Nonadditive) Join Decompositions

- **Claim 2: Preservation of Nonadditivity in Successive Decompositions**
  If a decomposition \( \text{DECOMP} = \{R_1, R_2, \ldots, R_m\} \) of \( R \) has the nonadditive (lossless) join property with respect to a set of functional dependency \( F \) on \( R \), and if a decomposition \( D_i = \{Q_1, Q_2, \ldots, Q_k\} \) of \( R_i \) has a nonadditive join property with respect to the projection of \( F \) on \( R_i \), then the decomposition \( D_2 = \{R_1, R_2, \ldots, R_i-1, Q_1, Q_2, \ldots, Q_k, R_{i+1}, \ldots, R_m\} \) of \( R \) has the nonadditive join property with respect to \( F \).

11.2 Algorithms for Relational Database Schema Design

11.2.1 Dependency-Preserving Decomposition into 3NF Schemas

- **Algorithm 11.2** Relational synthesis algorithm with dependency-preserving
  
  **Input:** A universal relation \( R \) and a set of functional dependencies \( F \) on the attributes of \( R \).
  
  **Output:** A dependency-preserving decomposition \( \text{DECOMP} = \{R_1, R_2, \ldots, R_n\} \) of \( R \) that all \( R_i \)'s in \( \text{DECOMP} \) are in 3NF.

  - 1. Find a minimal cover \( G \) for \( F \);
  
  - 2. For each left-hand-side \( X \) of a functional dependency that appears in \( G \), create a relation schema in \( \text{DECOMP} \) with attributes \( \{X \cup \{A_1\} \cup \{A_2\} \ldots \cup \{A_k\}\} \), where \( X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_k \) are the only dependencies in \( G \) with \( X \) as left-hand-side (\( X \) is the key of this relation);
  
  - 3. Place any remaining attributes (that have not been placed in any relation) in a single relation schema to ensure the attribute preservation property.

- Example:
  
  \( R = \{A, B, C, D, E, H\} \),
$F = \{AE \rightarrow BC, B \rightarrow AD, CD \rightarrow E, E \rightarrow CD, A \rightarrow E\}$.

Find a dependency-preserving decomposition $DECOMP = \{R_1, R_2, \ldots, R_n\}$ of $R$ such that each $R_i$ in $DECOMP$ is in 3NF.

- **Step 1:** A minimal cover $G = \{A \rightarrow B, A \rightarrow E, B \rightarrow A, CD \rightarrow E, E \rightarrow CD\}$ of $F$ is derived from algorithm 10.2.

- **Step 2:** Decompose $R$ to
  - $R_1 = \{A, B, E\}$ and $F_1 = \{A \rightarrow B, A \rightarrow E\}$
  - $R_2 = \{B, A\}$ and $F_2 = \{B \rightarrow A\}$
  - $R_3 = \{C, D, E\}$ and $F_3 = \{CD \rightarrow E\}$
  - $R_4 = \{E, C, D\}$ and $F_4 = \{E \rightarrow CD\}$

  Combine $R_3$ and $R_4$ into one relation schema
  - $R_5 = \{C, D, E\}$ and $F_5 = \{CD \rightarrow E, E \rightarrow CD\}$.

- **Step 3:** There is one attribute $H$ in $R - (R_1 \cup R_2 \cup R_5)$. Create another relation schema to contain this attribute.
  - $R_6 = \{H\}$ and $F_6 = \{\}$.  

  All relational schemas in the decomposition $DECOMP = \{R_1, R_2, R_5, R_6\}$ are in 3NF.

- Notice that the dependency are preserved: $\{F_1 \cup F_2 \cup F_5 \cup F_6\}^+ = F^+$.

- **Claim 3:** Every relation schema created by Algorithm 11.2 is in 3NF.

### 11.2.2 Lossless (Nonadditive) Join Decomposition into BCNF Schemas

- **Algorithm 11.3** Relational decomposition into BCNF relations with lossless join property
  
  **Input:** A universal relation $R$ and a set of functional dependencies $F$ on the attributes of $R$.

  - 1. Set $DECOMP = \{R\}$;
2. While there is a relation schema \( Q \) in \( DECOMP \) that is not in BCNF do

\[
\begin{align*}
\{ & \\
& \text{choose a relation schema } Q \text{ in } DECOMP \text{ that is not in BCNF;}
& \text{find a functional dependency } X \rightarrow Y \text{ in } Q \text{ that violates BCNF;}
& \text{replace } Q \text{ in } DECOMP \text{ by two relation schemas } (Q − Y) \text{ and } (X \cup Y);
\}
\]

\textbf{Why:} Since \((Q − Y) \cap (X \cup Y) \rightarrow (X \cup Y) − (Q − Y)\) is equivalent to \(X \rightarrow Y \in F^+\). By Property LJ1, the decomposition is lossless.

- Example: See Figure 10.11, 10.12, and Figure 10.13 (Fig 14.11, 14.12, 14.13 on e3).

- Example:

\( R = \{A, B, C\} \)

\( F = \{AB \rightarrow C, C \rightarrow B\} \)

- **Step 1:** Let \( DECOMP = \{\{A, B, C\}\}; \)

- **Step 2:** \( \{A, B, C\} \) in \( DECOMP \) that is not in BCNF;

  \( \text{Pick } \{A, B, C\} \text{ in } DECOMP; \)

  \( \text{Pick } C \rightarrow B \text{ in } \{A, B, C\} \text{ that violates BCNF; } \)

  \( \text{Replace } \{A, B, C\} \text{ in } DECOMP \text{ by } \{A, C\} \text{ and } \{B, C\}; \)

**Step 2:** \( DECOMP = \{\{A, C\}, \{B, C\}\} \) and both \( \{A, C\} \) and \( \{B, C\} \) are in BCNF;

Therefore, the decomposition \( DECOMP = \{\{A, C\}, \{B, C\}\} \) has the lossless join property.

(Try to use the **Algorithm 11.1** to test this decomposition)

- Another example:

\( R = \{A, B, C, D, E\} \)

\( F = \{AB \rightarrow CDE, C \rightarrow A, E \rightarrow D\} \)

- **Step 1:** Let \( DECOMP = \{\{A, B, C, D, E\}\}; \)

- **Step 2:** \( \{A, B, C, D, E\} \) in \( DECOMP \) that is not in BCNF;

  \( \text{Pick } \{A, B, C, D, E\} \text{ in } DECOMP; \)
Pick $C \rightarrow A$ in $\{A, B, C, D, E\}$ that violates BCNF;
Replace $\{A, B, C, D, E\}$ in $DECOMP$ by $\{B, C, D, E\}$ and $\{A, C\}$;

**Step 2:** $DECOMP = \{\{B, C, D, E\}, \{A, C\}\}$ and $\{B, C, D, E\}$ in $DECOMP$ that is not in BCNF;
Pick $\{B, C, D, E\}$ in $DECOMP$;
Pick $E \rightarrow D$ in $\{B, C, D, E\}$ that violate BCNF;
Replace $\{B, C, D, E\}$ in $DECOMP$ by $\{B, C, E\}$ and $\{D, E\}$;

**Step 2:** $DECOMP = \{\{A, C\}, \{B, C, E\}, \{D, E\}\}$ and all of them are in BCNF;
Therefore, the decomposition $DECOMP$ has the lossless join property.

(Try to use the **Algorithm 11.1** to test this decomposition)

- The same example, but try to pick a different FD first that violate BCNF

$R = \{A, B, C, D, E\}$
$F = \{AB \rightarrow CDE, C \rightarrow A, E \rightarrow D\}$

- **Step 1:** Let $DECOMP = \{\{A, B, C, D, E\}\}$;

- **Step 2:** $\{A, B, C, D, E\}$ in $DECOMP$ that is not in BCNF;
Pick $\{A, B, C, D, E\}$ in $DECOMP$;
Pick $E \rightarrow D$ in $\{A, B, C, D, E\}$ that violates BCNF;
Replace $\{A, B, C, D, E\}$ in $DECOMP$ by $\{A, B, C, E\}$ and $\{D, E\}$;

**Step 2:** $DECOMP = \{\{A, B, C, E\}, \{D, E\}\}$ and $\{A, B, C, E\}$ in $DECOMP$ that is not in BCNF;
Pick $\{A, B, C, E\}$ in $DECOMP$;
Pick $C \rightarrow A$ in $\{A, B, C, E\}$ that violate BCNF;
Replace $\{A, B, C, E\}$ in $DECOMP$ by $\{B, C, E\}$ and $\{A, C\}$;

**Step 2:** $DECOMP = \{\{A, C\}, \{B, C, E\}, \{D, E\}\}$ and all of them are in BCNF;
Therefore, the decomposition $DECOMP$ has the lossless join property.
The order of $FDs$ to be applied for decomposition does not matter.

### 11.2.3 Dependency-Preserving and Nonadditive (Lossless) Join Decomposition into 3NF Schemas
• **Algorithm 11.4** Relational synthesis into 3NF with dependency preservation and Nonadditive (lossless) join property

**Input:** A universal relation $R$ and a set of functional dependencies $F$ on the attributes of $R$.  

**Output:** A dependency-preserving and lossless-join decomposition $DECOMP = \{R_1, R_2, \ldots, R_n\}$ of $R$ that all $R_i$’s in $DECOMP$ are in 3NF.

1. Find a minimal cover $G$ for $F$ (use algorithm 10.2).

2. For each left-hand-side $X$ of a functional dependency that appears in $G$ create a relation schema in $DECOMP$ with attributes $\{X \cup \{A_1\} \cup \{A_2\} \ldots \cup \{A_k\}\}$, where $X \rightarrow A_1, X \rightarrow A_2, \ldots, X \rightarrow A_k$ are the only dependencies in $G$ with $X$ as left-hand-side ($X$ is the key of this relation)

3. If none of the relation schemas in $DECOMP$ contains a key of $R$, then create one more relation schema in $D$ that contains attributes that form a key of $R$.

• Example:

$R = \{A, B, C, D, E, H\}$,  
$F = \{AE \rightarrow BC, B \rightarrow AD, CD \rightarrow E, E \rightarrow CD, A \rightarrow E\}$.

Find a dependency-preserving and lossless-join decomposition $DECOMP = \{R_1, R_2, \ldots, R_n\}$ of $R$ such that each $R_i$ in $DECOMP$ is in 3NF.

1. **Step 1:** A minimal cover $G = \{A \rightarrow B, \ A \rightarrow E, \ B \rightarrow A, \ CD \rightarrow E, \ E \rightarrow CD\}$ of $F$ is derived from algorithm 10.2.

2. **Step 2:** Decompose $R$ to

   $R_1 = \{A, B, E\}$ and $F_1 = \{A \rightarrow B, \ A \rightarrow E\}$

   $R_2 = \{B, A\}$ and $F_2 = \{B \rightarrow A\}$

   $R_3 = \{C, D, E\}$ and $F_3 = \{CD \rightarrow E\}$

   $R_4 = \{E, C, D\}$ and $F_4 = \{E \rightarrow CD\}$

   Combine $R_3$ and $R_4$ into one relation schema

   $R_5 = \{C, D, E\}$ and $F_5 = \{CD \rightarrow E, \ E \rightarrow CD\}$.

3. **Step 3:** $AH$ and $BH$ are candidate keys of $R$, and neither of them appear in
Create another relation schema
\[ R_6 = \{ A, H \} \text{ and } F = \{ \} \]
Then, all relational schemas in the decomposition \( DECOMP = \{ R_1, R_2, R_5, R_6 \} \) are in 3NF.

**Dependency preservation:** \( \{ F_1 \cup F_2 \cup F_5 \cup F_6 \}^+ = F^+ \)

**Lossless join property:** Try to use algorithm 11.1 to test this decomposition.

### Appendix: An Example to Summarize Functional Dependencies and Normal Forms

- Let a relation schema \( R = \{ A, B, C \} \) with functional dependency set
  \[ F = \{ A \rightarrow BC, \ AB \rightarrow C, \ C \rightarrow B \} \]

  - **Question#1:** Is \( G = \{ A \rightarrow B, \ AB \rightarrow C, \ C \rightarrow B \} \) equivalent to \( F \)?
    1. Does \( G \models A \rightarrow BC \)?
       Since \( A^+_G = ABC \supseteq BC \Rightarrow Yes \).
    2. Does \( F \models A \rightarrow B \)?
       Since \( A^+_F = ABC \supseteq B \Rightarrow Yes \).
       Both (i) and (ii) return true, then \( G \equiv F \).

  - **Question#2:** Find a minimal cover \( G \) of \( F \).
    1. Let \( G_1 = \{ A \rightarrow B, \ A \rightarrow C, \ AB \rightarrow C, \ C \rightarrow B \} \)
    2. Check for left redundancy.
       1) Is \( A \) redundant in \( AB \rightarrow C \) of \( G_1 \)?
          Let \( G_2 = \{ A \rightarrow B, \ A \rightarrow C, \ B \rightarrow C, \ C \rightarrow B \} \)
          Does \( G_1 \models B \rightarrow C \)? Since \( B^+_G = B \not\supseteq C \Rightarrow No \)
          Does \( G_2 \models AB \rightarrow C \)?
             Thus, \( A \) is **needed**.
       2) Is \( B \) redundant in \( AB \rightarrow C \) of \( G_1 \)?
Let $G_3 = \{A \rightarrow B, \ A \rightarrow C, \ C \rightarrow B\}$

$G_1$ covers $G_3$ Since $G_1 \supseteq G_3$

Does $G_3 \models AB \rightarrow C$? Since $AB^+_G = ABC \supseteq C \Rightarrow Yes$.

Thus, $B$ is redundant.

(iii) Check for FDs redundancy.

1) Is $A \rightarrow B$ redundant in $G_3$?

Let $G_4 = \{A \rightarrow C, C \rightarrow B\}$

Does $G_4 \models A \rightarrow B$? $A^+_G = ACB \supseteq B \Rightarrow Yes$.

Thus, $A \rightarrow B$ is redundant.

2) Is $A \rightarrow C$ redundant in $G_4$?

Let $G_5 = \{C \rightarrow B\}$

Does $G_5 \models A \rightarrow C$? $A^+_G = A \not\supseteq B \Rightarrow No$.

Thus, $A \rightarrow C$ is needed.

Conclusion, $G = G_1 = \{A \rightarrow C, \ C \rightarrow B\}$ is a minimal cover of $F$.

- Question#3: Find all possible candidate keys of $R$.

  (i) One-attribute keys:

  $A^+_F = ABC = R \Rightarrow A$ is a superkey (thus, a candidate key).

  $B^+_F = B \neq R$

  $C^+_F = CB \neq R$

  (ii) Two-attribute keys: We only need to check $BC$ because all other combination will contain $A$.

  $BC^+_F = BC \neq R$

  (iii) Three-attribute keys: There is only one possible combination $ABC$. Since $A$ is a key, $ABC$ is not a key.

  Conclusion, There is only one candidate key $A$.

- Question#4: For the general definition, is $R$ in 2NF? If not, decompose it to 2NF.
There is only one key with length one. Therefore, no partial dependency exist, i.e., \( R \) is in 2NF.

- Question#5: For the general definition, are all relation schemas of the resulting decomposition from Q4 in 3NF? If not, decompose them to 3NF.

  (i) For \( A \rightarrow BC \), \( A \) is a superkey (ok.)

  (ii) For \( AB \rightarrow C \), \( AB \) is a superkey (ok.)

  (iii) For \( C \rightarrow B \), \( C \) is not a superkey and \( B \) is not a prime attribute (not ok.)

  Decomposition:

  \[
  R(ABC) \Rightarrow \begin{cases} 
  R_1(AC) & F_1 = \{ A \rightarrow C \} \\
  R_2(BC) & F_2 = \{ C \rightarrow B \}
  \end{cases}
  \]

- Question#6: For the general definition, are all relation schemas of the resulting decompositions from Q5 in BCNF? If not, decompose them to BCNF.

  Both \( R_1 \) and \( R_2 \) are in BCNF.

- Question#7: Does the resulting decompositions from previous questions have dependency-preserving and lossless-join properties?

  (i) Check for lossless-join property.

  | \( R_1 \) | \( A \) | \( B \) | \( C \) |
  |---|---|---|
  | \( a_1 \) | \( b_{12} \) | \( a_2 \) | \( a_3 \) |
  | \( b_{21} \) | \( a_2 \) | \( a_3 \) |

  For \( A \rightarrow BC \) No change.

  \( AB \rightarrow C \) No change.

  \( C \rightarrow B \) change \( b_{12} \rightarrow a_2 \).

  We have a row with all \( a_i \) symbols ⇒ It has lossless-join property.

  (ii) Check for dependency-preserving property.

  Does \( (F_1 \cup F_2) \equiv F \)?

  1) Does \( F \models A \rightarrow C \)? Yes.

  2) Does \( F_1 \cup F_2 \models A \rightarrow BC, AB \rightarrow C \)? Yes.

  Therefore, it has the dependency-preserving property.