Chapter 7: Relational Database Design

- Pitfalls in Relational Database Design
- Decomposition
- Normalization Using Functional Dependencies
- Normalization Using Multivalued Dependencies
- Normalization Using Join Dependencies
- Domain-Key Normal Form
- Alternative Approaches to Database Design
Pitfalls in Relational Database Design

- Relational database design requires that we find a “good” collection of relation schemas. A bad design may lead to
  - Repetition of information.
  - Inability to represent certain information.

- Design Goals:
  - Avoid redundant data
  - Ensure that relationships among attributes are represented
  - Facilitate the checking of updates for violation of database integrity constraints
Example

- Consider the relation schema:

  \[
  \text{Lending-schema} = (\text{branch-name}, \text{branch-city}, \text{assets}, \\
  \text{customer-name}, \text{loan-number}, \text{amount})
  \]

- Redundancy:
  - Data for \text{branch-name}, \text{branch-city}, \text{assets} are repeated for each loan that a branch makes
  - Wastes space and complicates updating

- Null values
  - Cannot store information about a branch if no loans exist
  - Can use null values, but they are difficult to handle
Decomposition

- Decompose the relation schema \textit{Lending-schema} into:

  \textit{Branch-customer-schema} = (\textit{branch-name, branch-city, assets, customer-name})

  \textit{Customer-loan-schema} = (\textit{customer-name, loan-number, amount})

- All attributes of an original schema \((R)\) must appear in the decomposition \((R_1, R_2)\):

  \[ R = R_1 \cup R_2 \]

- Lossless-join decomposition.
  For all possible relations \(r\) on schema \(R\)

  \[ r = \Pi_{R_1}(r) \bowtie \Pi_{R_2}(r) \]
Example of a Non Lossless-Join Decomposition

- Decomposition of \( R = (A, B) \)
  \( R_1 = (A) \quad R_2 = (B) \)

\[
\begin{array}{c|c}
A & B \\
\hline
\alpha & 1 \\
\alpha & 2 \\
\beta & 1 \\
r & \end{array}
\quad \begin{array}{c}
A \\
\hline
\alpha \\
\beta \\
\end{array}
\quad \begin{array}{c}
B \\
\hline
1 \\
2 \\
\end{array}
\]

- \( \Pi_A(r) \times \Pi_B(r) \)

\[
\begin{array}{c|c}
A & B \\
\hline
\alpha & 1 \\
\alpha & 2 \\
\beta & 1 \\
\beta & 2 \\
\end{array}
\]
Goal — Devise a Theory for the Following:

- Decide whether a particular relation $R$ is in “good” form.
- In the case that a relation $R$ is not in “good” form, decompose it into a set of relations $\{R_1, R_2, \ldots, R_n\}$ such that
  - each relation is in good form
  - the decomposition is a lossless-join decomposition
- Our theory is based on:
  - functional dependencies
  - multivalued dependencies
Normalization Using Functional Dependencies

When we decompose a relation schema $R$ with a set of functional dependencies $F$ into $R_1$ and $R_2$ we want:

- **Lossless-join decomposition**: At least one of the following dependencies is in $F^+$:
  
  - $R_1 \cap R_2 \rightarrow R_1$
  - $R_1 \cap R_2 \rightarrow R_2$

- **No redundancy**: The relations $R_1$ and $R_2$ preferably should be in either Boyce-Codd Normal Form or Third Normal Form.

- **Dependency preservation**: Let $F_i$ be the set of dependencies in $F^+$ that include only attributes in $R_i$. Test to see if:
  
  - $(F_1 \cup F_2)^+ = F^+$

Otherwise, checking updates for violation of functional dependencies is expensive.
Example

• \( R = (A, B, C) \)
  \[ F = \{ A \rightarrow B, B \rightarrow C \} \]

• \( R_1 = (A, B), \quad R_2 = (B, C) \)
  – Lossless-join decomposition:
    \[ R_1 \cap R_2 = \{ B \} \text{ and } B \rightarrow BC \]
  – Dependency preserving

• \( R_1 = (A, B), \quad R_2 = (A, C) \)
  – Lossless-join decomposition:
    \[ R_1 \cap R_2 = \{ A \} \text{ and } A \rightarrow AB \]
  – Not dependency preserving
    (cannot check \( B \rightarrow C \) without computing \( R_1 \Join R_2 \))
A relation schema $R$ is in BCNF with respect to a set $F$ of functional dependencies if for all functional dependencies in $F^+$ of the form $\alpha \rightarrow \beta$, where $\alpha \subseteq R$ and $\beta \subseteq R$, at least one of the following holds:

- $\alpha \rightarrow \beta$ is trivial (i.e., $\beta \subseteq \alpha$)
- $\alpha$ is a superkey for $R$
Example

- \( R = (A, B, C) \)
  \[ F = \{ A \rightarrow B, \quad B \rightarrow C \} \]
  Key = \{ A \}

- \( R \) is not in BCNF

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

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

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

- \( R = (\text{branch-name}, \text{branch-city}, \text{assets}, \text{customer-name}, \text{loan-number}, \text{amount}) \)

- \( F = \{ \text{branch-name} \rightarrow \text{assets} \text{ branch-city} \}

- \( \text{loan-number} \rightarrow \text{amount} \text{ branch-name} \} \)

- \( \text{Key} = \{ \text{loan-number}, \text{customer-name} \} \)

- Decomposition
  - \( R_1 = (\text{branch-name}, \text{branch-city}, \text{assets}) \)
  - \( R_2 = (\text{branch-name}, \text{customer-name}, \text{loan-number}, \text{amount}) \)
  - \( R_3 = (\text{branch-name}, \text{loan-number}, \text{amount}) \)
  - \( R_4 = (\text{customer-name}, \text{loan-number}) \)

- Final decomposition
  \( R_1, R_3, R_4 \)
It is not always possible to get a BCNF decomposition that is dependency preserving.

**Example:**

- $R = (J, K, L)$
- $F = \{JK \rightarrow L, L \rightarrow K\}$

Two candidate keys = $JK$ and $JL$

- $R$ is not in BCNF
- Any decomposition of $R$ will fail to preserve $JK \rightarrow L$