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

Chapter 7: Relational Database Design
Integrity constraints
- Facilitate the checking of updates for violation of database
- Ensure that relationships among attributes are represented
- Avoid redundant data

Design Goals:
- Imability to represent certain information.
- Repetition of information.

Collection of relation schemes. A bad design may lead to

Relational database design requires that we find a "good"

Pitfalls in Relational Database Design
- Can use null values but they are difficult to handle
- Can not store information about a branch if no loans exist
  - Null values
  - Wastes space and complicates updating
- Each loan that a branch makes
- Data for branch-name, branch-city, assets are repeated for
  - Redundancy

Consider the relation schema:

```
<table>
<thead>
<tr>
<th>customer-name</th>
<th>loan-number</th>
<th>amount</th>
</tr>
</thead>
</table>
```

Example schema: (branch-name, branch-city, assets,)

```sql
 Len@nding-schema = (branch-name, branch-city, assets,)
```
\[(\mathcal{D}) \bigotimes (\mathcal{H}^1_{R_2}) = \mathcal{H}
\]

For all possible relations \( \mathcal{R} \) on schema \( \mathcal{H} \):
- Lossless-join decomposition.

\[R_2 \cap R_1 = R
\]

decomposition \((R_1, R_2)\):
- All attributes in an original schema \( \mathcal{H} \) must appear in the.
  - Customer-loan-schema = (customer-name, loan-number, amount)
    - (assets, customer-name)
    - Branch-customer-schema = (branch-name, branch-city)

Decompose the relation schema Loan into:

\textbf{Decomposition}
Example of a Non Lossless-Join Decomposition
multivalued dependencies

functional dependencies

Our theory is based on:

the decomposition is a lossless-join decomposition

each relation is in good form

it into a set of relations \( \{ R_1, R_2, \ldots, R_n \} \) such that

In the case that a relation \( R \) is not in "good" form, decompose

decide whether a particular relation \( R \) is in "good" form.
Normalization Using Functional Dependencies

Dependences is expensive.
Otherwise checking updates for violation of functional dependencies is expensive.

\[ \text{Ext} = (F_1 \cup F_2) - \text{Ext} \]

Test to see if \( \text{Ext} \) contains only attributes from \( R' \).

- Dependency preservation: Let \( R' \) be the set of dependences in either Boyce-Codd Normal Form or Third Normal Form,

- No redundancy: The relations \( R_1 \) and \( R_2 \) preferably should be:

\[ R_2 \leftarrow R_1 \cup R_2 \]

\[ R_1 \leftarrow R_1 \cup R_2 \]

When we decompose a relation schema \( R \) with a set of functional dependences
\((R_2 \times R_1) \not\subseteq B\) without computing \(R_1\)

- **Not dependency preserving**

\[
B \subseteq A \quad \text{and} \quad \{A\} = R_2 \cup R_1
\]

- **Lossless-join decomposition:**

\[
(C, A') = R_2 \quad \text{and} \quad \{A\} = R_1 \quad \bullet
\]

- **Dependency preserving**

\[
C \subseteq B \quad \text{and} \quad \{B\} = R_2 \cup R_1
\]

- **Lossless-join decomposition:**

\[
(C, B') = R_2 \quad \text{and} \quad \{C \leftarrow B, B' \leftarrow A\} = R
\]

\[
(C, B') = R \quad \bullet
\]

**Example**
A relation schema \( R \) is in Boyce-Codd Normal Form (BCNF) if for all functional dependencies of the form \( a \rightarrow B \), where \( a \subseteq R \), and \( a \subseteq F^+ \), at least one of the following holds:

- \( a \) is a superkey for \( R \)
- \( a \subseteq B \), i.e., \( B \) is trivial.
Dependency preserving

- Lossless-join decomposition

- $R_1$ and $R_2$ in BCNF

$\langle C \bowtie B, C \rangle = \Pi_{R_2}(\Pi_{R_1}(\Pi_{\forall}(B, C)))$

Decomposition

- $R$ is not in BCNF

- Key

$\{\forall\} = K$

$\{C \leftarrow B$

$B \leftarrow \forall\} = \Pi$

$\langle \forall, B, C \rangle = R$
Note: each $R_i$ is in BCNF, and decomposition is lossless-join.

```plaintext
else done := true;
end

\[
\text{result} := (R - \text{result}) \cap (g - (g \cap (R_i \neq R)))
\]

and $\emptyset = g \cup R_i$ such that $R_i$ is not in $R$.

Let $a$ be a nontrivial functional dependency.

then begin

if (there is a schema $R_i$ in result that is not in BCNF)

while (not done) do

Compute $R_i^{+}$.

done := false;

\{R\} := result
```

BCNF Decomposition Algorithm
Final decomposition

(\text{customer-name, loan-number}) = R^4 - 
(\text{branch-name, loan-number, amount}) = R^3 - 
(\text{branch-name, customer-name, loan-number, amount}) = R^2 - 
(\text{branch-name, branch-city, assets}) = R^1 - 

Decomposition

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

\{\text{loan-number} \leftarrow \text{amount, branch-name} \leftarrow \text{assets, branch-city}\} = \mathcal{F}

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

\text{Example of BCNF Decomposition}
\( \ell I \leftarrow \ell K \)

Any decomposition of \( R \) will fail to preserve

\( R \) is not in BCNF

Two candidate keys = \( \ell K \) and \( \ell I \)

\[ \{ \ell Y \leftarrow \ell I \]  
\[ \ell I \leftarrow \ell K \} = \mathcal{F} \]

\( (\ell I, \ell K, \ell J) = \mathcal{F} \)

It is not always possible to get a BCNF decomposition that is

BCNF and Dependency Preservation
If a relation is in $B$CNF it is in 3$NF$ (since in $B$CNF one of the
first two conditions above must hold).

- Each attribute $A$ in $R$ is contained in a candidate key.
- $A$ is a superkey for $R$.
- $(A \in \beta, \beta \text{ is trivial (i.e., trivial)} \iff A \in \beta$)

At least one of the following holds:

$$+ \beta \in \beta \iff A$$

A relation schema $R$ is in third normal form (3$NF$) if for all:

**Third Normal Form**
dependency preserving
- lossless-join decomposition
- each relation schema $R_i$ is in 3NF

Relation schemas $\{R_1, \ldots, R_n\}$ such that:

Algorithm to decompose a relation schema $R$ into a set of

$R$ is contained in a candidate key

$R$ is a superkey

$R$ is in 3NF

If and two candidate keys:

$\{K \leftarrow T, T \leftarrow KF\} = H$

$(T, K, T) = H$ - Example

3NF (Cont.)
\[ \text{return } (R_1, R_2, \ldots, R_n) \]

end

\[ H_i = \text{any candidate key for } R_i \]

\[ + \] \[ H_i \]

then begin

\[ a \text{ = any candidate key for } R_i \]

\[ I \supseteq \] \[ R \subseteq I \supseteq \] \[ \text{none of the schemes } R \text{ contains } I \supseteq \] \[ \text{none of the schemes } R \]

end

\[ H_i \text{ \&} \]

\[ + \] \[ H_i \]

then begin

\[ a \text{ contains } I \supseteq \] \[ R \subseteq I \supseteq \] \[ \text{none of the schemes } R \text{ contains } I \supseteq \] \[ \text{none of the schemes } R \]

for each functional dependency \( a \leftarrow b \) in \( R \) do

\[ 0 \]

\[ = \]

\[ H_i \]

let \( H_i \) be a canonical cover for \( R_i \)

\[ \text{3NF Decomposition Algorithm} \]
\{ \text{customer-name, branch-name} \}

The key is:

\text{customer-name, branch-name} \rightarrow \text{banker-name}

\text{banker-name} \rightarrow \text{branch-name, office-number}

The functional dependencies for this relation schema are:

\text{banker-name, office-number}

\text{Banker-info-schema = (branch-name, customer-name,}

Relation schema:

\text{Example}
process.

Since Banker-schema contains a candidate key for

\[ \text{Banker-schema} = \langle \text{customer-name, branch-name, } \rangle \]

\[ \text{Banker-office-schema} = \langle \text{banker-name, branch-name, } \rangle \]

The for loop in the algorithm causes us to include the

Applying 3NF to Banker-info-schema
It may not be possible to preserve dependencies – the decomposition is lossless.

BCNF and 3NF

It is always possible to decompose a relation into relations in

dependencies are preserved – the decomposition is lossless.

3NF and 3NF
Consider the following relation

\[
\{ X \leftarrow T \\
T \leftarrow XJ \} = E
\]

\[(T, J', K') = R \]

Comparison of BCNF and 3NF (cont.)
- Dependency preservation.
- Lossless join.
- 3NF.
- BNF.

If we cannot achieve this, we accept:
- Dependency preservation.
- Lossless join.
- BNF.

Goal for a relational database design is:
Normalization Using Multivalued Dependencies

matter who teaches it).

no teachers any one of which can be the course’s instructor, and

The database is supposed to list for each course the set of

and if is a required textbook for c

classess(course, teacher, book)

Consider a database

sufficiency normalized

There are database schemas in BCNF that do not seem to be
(database, Sara, Ulman)
(database, Sara, Korth)
teach database, two tuples need to be inserted
• Insertion anomalies – i.e., if Sara is a new teacher that can
  book (is the only key, and therefore the relation is in BCNF
Since there are no non-trivial dependencies, (course, teacher,

<table>
<thead>
<tr>
<th>classes</th>
<th>Shaw</th>
<th>Silberschatz</th>
<th>Shaw</th>
<th>Silberschatz</th>
<th>Ulman</th>
<th>Korth</th>
<th>Ulman</th>
<th>Korth</th>
<th>Ulman</th>
<th>Korth</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Shaw</td>
<td></td>
<td>Shaw</td>
<td></td>
<td>Ulman</td>
<td></td>
<td>Ulman</td>
<td></td>
<td>Ulman</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
|         |      |                |      |                |       |       |       |       |       |�

| book | teacher | course |

(Shaw, Silberschatz, Ulman, Korth, database, systems, course, teacher)
We shall see that these two relations are in Fourth Normal Form (4NF).

Therefore, it is better to decompose classes into:

```
text
  Shaw  operating systems
  Silberschatz  operating systems
  Ulman  database
  Korth  database
|
book  course


teaches
  Jim  operating systems
  Avi  operating systems
  Sudarshan  database
  Hank  database
  Avi  database
|
teacher  course
```
\[ [\theta - 2]_4^t = [\theta - 4]_4^t \]
\[ [\theta]_2^t = [\theta]_4^t \]
\[ [\theta - 2]_3^t = [\theta - 4]_3^t \]
\[ [\theta]_1^t = [\theta]_3^t \]
\[ [\nu]_4^t = [\nu]_3^t = [\nu]_2^t = [\nu]_1^t \]

In \( t \) such that:

there exist tuples \( t_3 \) and \( t_4 \) in \( t \) such that \( t_3^t \nu = [\nu]_3^t \)

for all pairs of tuples \( t_1 \) and \( t_2 \) in any legal relation \( \mathcal{L}(R) \).

Let \( R \) be a relation schema and let \( \theta \) and \( \bar{R} \) and \( \theta \subseteq \bar{R} \). The

**Multiivalued Dependencies (MVDs)**

\[ \theta \leftrightarrow \bar{R} \]
<table>
<thead>
<tr>
<th>$u_D$ ... $I + c_D$</th>
<th>$c_q$ ... $I + c_q$</th>
<th>$c_D$ ... $I$</th>
<th>$\mathfrak{t}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>$u_q$ ... $I + c_q$</td>
<td>$c_D$ ... $I + c_D$</td>
<td>$c_D$ ... $I$</td>
<td>$\mathfrak{t}$</td>
</tr>
<tr>
<td>$u_D$ ... $I + c_D$</td>
<td>$c_q$ ... $I + c_q$</td>
<td>$c_D$ ... $I$</td>
<td>$\mathfrak{t}$</td>
</tr>
<tr>
<td>$\mathfrak{g}$</td>
<td>$\mathfrak{r}$</td>
<td>$\mathfrak{r}$</td>
<td>$\mathfrak{t}$</td>
</tr>
<tr>
<td>$\mathfrak{g}$</td>
<td>$\mathfrak{r}$</td>
<td>$\mathfrak{r}$</td>
<td>$\mathfrak{t}$</td>
</tr>
</tbody>
</table>

$\mathfrak{g}$ $\leftrightarrow$ $\mathfrak{a}$ $\mathfrak{r}$ $\mathfrak{r}$ $\mathfrak{r}$ $\mathfrak{t}$ $\mathfrak{t}$ $\mathfrak{t}$

**Table: Representation of $\mathfrak{a}$**

[MID (cont.)]
\[
M \leftrightarrow \Lambda \iff Z \leftrightarrow \Lambda \text{ are identical if follows that}
\]

Note that since the behavior of \( M \) and \( Z \) are identical, it follows that

\[
\forall t \in \mathcal{L}, \exists m_2, m_1, z_1, z_2 \text{ such that } t \in Z \text{ and } \exists \}
\]

then

\[
\forall t \in \mathcal{L}, \exists m_2, m_1, z_1, z_2 \text{ such that } t \in Z \text{ and } \exists \}
\]

if and only if for all possible relations \( \mathcal{R} \)

\[
(H) \quad (Z \text{ multi-determines } \Lambda) \quad Z \leftrightarrow \Lambda \text{ we say that }
\]

\[
M \downarrow Z \downarrow \Lambda \text{ partitioned into } 3 \text{ non-empty subsets, }
\]

Let \( R \) be a relation schema with a set of attributes that are

Example
The claim follows.

\[ Z = Z \] (in above notation)

Indeed we have \( Z \leftarrow \lambda \) then \( Z \leftarrow \lambda \)

Note: Each other:

- The above formal definition is supposed to formalize the notion of book, course, teacher, that given a particular value of \( \lambda \) has associated with it a set of values of \( M \) and these two sets are in some sense independent of each other.

Example (cont.)
If a relation $r$ fails to satisfy a given multivalued dependency, we can construct a relation $r'$ that does satisfy the multivalued dependency by adding tuples to $r$. Given set of functional and multivalued dependencies.

This concerns ourselves only with relations that satisfy a given set of functional and multivalued dependencies.

1. To test relations to determine whether they are legal under multivalued dependencies in two ways:

   • Use of Multivalued Dependencies

   •
Theory of Multivalued Dependencies

- Sound and complete inference rules for functional and multivalued dependencies
- Logically implied by $D$
- Let $D$ denote a set of functional and multivalued dependencies.

1. **Reflexivity Rule.** If $\forall a$ holds, then $a \subseteq a$ holds.

2. **Augmentation Rule.** If $a \subseteq \gamma$ holds, then $a \gamma \subseteq \gamma$ holds.

3. **Transitivity Rule.** If $a \gamma \subseteq \delta$ holds and $\delta \gamma \subseteq \nu$ holds, then $a \nu \subseteq \gamma$ holds.

["multivalued dependencies"]
then \( \forall \) holds.

there is a \( \forall \) such that \( \forall \) and \( \forall \) and \( \forall \) holds and

8. Coalescence rule. If \( \forall \) holds, then \( \forall \) holds.

7. Replication rule. If \( \forall \) holds, then \( \forall \) holds.

6. Multivalued transitivity rule. If \( \forall \) holds, then \( \forall \) holds.

5. Multivalued augmentation rule. If \( \forall \) holds, then \( \forall \) holds.

4. Complementation rule. If \( \forall \) holds, then

Theory of Multivalued Dependencies (Cont.)
\textbf{Difference rule.} If a holds, then \( \emptyset \not\rightarrow a \) holds.
\textbf{Intersection rule.} If a holds, then \( \emptyset \not\rightarrow a \) holds.
\textbf{Multiplication rule.} If a holds, then \( \emptyset \not\rightarrow a \) holds.

\begin{itemize}
  \item the following rules (proved using rules 1–8).
\end{itemize}

We can simplify the computation of the closure of \( D \) by using

\begin{itemize}
  \item the computation of the Computation of the Computation of \( D' \)
Since $IH = B - B$.

Since $IH$, the multi-Valued transitivity rule (6) implies that $A$.

Since $A$, $B \leftrightarrow A - IH$. 

Since $A$, $B \leftrightarrow A - CGHI$. 

Since $B$, the complementation rule (4) implies that $A$.

Some members of $D^+$ are:

{ $H \leftarrow GC$ 
$IH \leftarrow B$ 
$B \leftarrow A$. $D = (A, B, C, H, I)$
Since $\text{CGHI} - \text{HI} = \text{CG}$, $A \rightarrow \text{CG}.$

By the difference rule, $A \rightarrow \text{CGHI} - \text{HI}$

$\text{HI} \leftarrow A \text{ CGHI and } \text{HI} \leftarrow A \text{ CG.}

We conclude that $H \leftarrow B - \text{CG, } \land \text{ being } H. \text{ Since } \emptyset \notin \text{HI and } \text{CG} \cap \text{HI} = \emptyset, \text{ the coalescence rule is satisfied with } a \text{ being } B, \text{ apply the coalescence rule (8)}; B \text{ HI holds.}

\begin{align*}
\text{HI} & \leftarrow B - H.
\end{align*}

Some members of $D^+_\text{cont.}$.
If a relation is in 4NF it is in BCNF

\[ R = \bigcap R' \cap R' \subseteq R \quad \text{if and only if} \quad R' \text{ is trivial} \]

If a relation is in 4NF it is in BCNF

- \( a \) is a superkey for schema \( R \)

A relation schema \( R \) is in 4NF with respect to a set of dependencies in \( D \) if for all multivalued functional and multivalued dependencies in \( D \) at least one of the following holds:

- \( R' \subseteq R \), where \( a \leftrightarrow \) where \( a \), and
- \( R' \subseteq R \)
Note: each \( R' \) is in 4NF, and decomposition is lossless-join.

```
else done := true;
end

\( \emptyset = \emptyset \cap (R') \cap (R - \text{result}) =: \text{result} \)

\( \emptyset = \emptyset \cup a \) and \( a \) is not in \( \mathcal{P} \) and \( a \) is not in \( \mathcal{F} \) such that

Let \( a \) be a nontrivial multivalued

then begin

(there is a schema \( R' \) in \( \text{result} \) that is not in \( \mathcal{P} \))

while \( \) not done \( ) \) do

compute \( \mathcal{P} \) + \( \mathcal{F} \);

done := false;

\{ R \} := \text{result}
```

### 4NF Decomposition Algorithm
(Rg is in 4NP)
(Rg)
(Rg is in 4NP)
(Rg is not in 4NP)
(Rg is not in 4NP)
(Rg is in 4NP)

\[ (\forall, A, H) = \gamma Rg (I, A, H) = \gamma Rg (I) \]
\[ (\forall, A, H) = \gamma Rg (I, A, H) = \gamma Rg (I) \]
\[ (\forall, A, H) = \gamma Rg (I, A, H) = \gamma Rg (I) \]

\[ I \leftrightarrow A, H, I \leftrightarrow A, H \leftrightarrow B \leftrightarrow B \leftrightarrow A \text{ and } B \leftrightarrow A \]

\[ \{ H \leftarrow C, C \}
IH \leftrightarrow B
B \leftrightarrow A \} = B

\[ \text{Example} \]
(even on just the multivalued dependencies)

Decomposition into 4NF may not be dependency preserving

\[ \Pi_x R \] for all \( x \in D \)

[1] satisfies \( R \) and for which \( \forall \Pi_x R \) that exists a relation \( R \) such

\[ \Pi_x R \] for every set of relations \( \Pi_x R \) such

\[ \Pi_x R \] for every set of relations \( \Pi_x R \) such

The decomposition is dependency-preserving with respect to \( D \)

\[ +D \]

where \( a \in H \) and \( a \) is in \( D \)

\[ H \cup g \]

\[ \Pi_x R \] for every set of relations \( \Pi_x R \) such

The restriction of \( D \) to \( R \) is the set \( D \) consisting of

both functional and multivalued dependencies.

Let \( R^1, R^2, \ldots, R^n \) be a decomposition of \( R \) and \( D \) a set of

Multi-valued Dependancy Preservation
Any multi-valued dependency.

However, there are join dependencies that are not equivalent to

\((\rho \cap a, \rho - \mathcal{R} \cap a)\) to \(a^*)\)

dependency \(\mathcal{R} \cup \mathcal{R}^2\) to \(\mathcal{R}^*\). Conversely, a \(\mathcal{R}\) is equivalent to the multi-valued dependency \(\mathcal{R}^1, \mathcal{R}^2\) is equivalent to the multi-valued dependency

A join dependency is trivial if one of the \(\mathcal{R}\) is \(\mathcal{R}^i\) itself.

\((\mathcal{R}^1 \cap \mathcal{R}^2 \cap \ldots \cap \mathcal{R}^i)\) \(\mathcal{R}^i \cap \mathcal{R}^i \cap \mathcal{R}^i = \mathcal{R}^i\)

relation \(\mathcal{R}^i\) satisfies the join dependency \(\mathcal{R}^1, \mathcal{R}^2, \ldots, \mathcal{R}^i\) \(\mathcal{R}^i\) \(\mathcal{R}^i\) \(\mathcal{R}^i\) decomposition of \(\mathcal{R}\) if \(\mathcal{R} = \mathcal{R}^1 \cap \mathcal{R}^2 \cap \ldots \cap \mathcal{R}^i\). We say that a

Let \(\mathcal{R}\) be a relation schema and \(\mathcal{R}^1, \mathcal{R}^2, \ldots, \mathcal{R}^i\) be a

a lossless-join decomposition.

A join dependency constitutes the set of legal relations over a

Normalization Using Join Dependencies
• Every P-JNF schema is also in 4NF.

Since every multi-valued dependency is also a join dependency,

- Every R is a superkey for R.
  
  - Every R is a superkey for R.

at least one of the following holds:

\[
R \subseteq R_1 \cap R_2 \cap \ldots \cap R_n
\]

and R = R_1 \cap R_2 \cap \ldots \cap R_n

where each R is in P-JNF. If for all join dependencies in D+, any relation schema R is in P-JNF with respect to a set D of

\[
\text{Project-Join Normal Form (P-JNF)}
\]
three schemas specified by the join dependency:

\( \text{loan-info-schema = (branch-name, customer-name, } \) \\
(\text{loan-number, amount)} \) –

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

(\text{loan-number, branch-name) \) –

Each loan has one or more customers, is in one or more

Consider \( \text{loan-info-schema = (branch-name, customer-name, } \) \\
\text{loan-number, amount)} \) –

Example
Insert C.

Imply C.

Constraint for R. Schema R is in DKNF if D \cap K is logically consistent for a relation schema R. Let C denote the general constraint for a set or domain constraints and let K be a set or key.

Let D be a set of all relations on a given schema.

General constraint. A general constraint is a predicate on dependencies but not all functional dependencies are key dependencies.

All key declarations are functional schema R (X \rightarrow Y). The key declaration Key (Y) requires that Y be a superkey for the relation declaration. Let Y be a relation schema with Y \subseteq R.

Key declaration. The X value of all tuples be values in dom. A set of values. The domain declaration dom requires that dom \subseteq dom requires that dom A be an attribute, and let dom be dom.
– The balance is greater than 2500.
– The account number begins with 9.

Each account:

Domain constraints for `special-act-schema` require that for:

\[
\text{special-act-schema} = (\text{branch-name}, \text{account-name, account-number, balance})
\]

\[
\text{regular-act-schema} = (\text{branch-name}, \text{account-name, account-number, balance})
\]

\[
\text{DKNP design:}
\]

\[
\text{then } t[\text{balance}] \geq 2500.
\]

General constraint: "If the first digit of \( t[\text{account-number}] \) is 9, \$2500. accounts whose account-number begins with the digit 9 are special high-interest accounts with a minimum balance of.

Example
• to D, R, and C.

Schema R is in PNF if and only if it is in DKNF with respect to nontrivial functional dependencies in F of the form a → R.

• Let the set K of key constraints be those dependencies in C that join dependencies of F are the general constraints of C be a set of functional r \subseteq \text{dom}(A')

• Then all domain constraints are of the form denote the domain of attribute A; and let all those domains be A = (A', A_2', \ldots, A_n') be a relation schema. Let DNF.
collection of normal form schemes from a given set of attributes.

If dangling tuples are allowed in the database, instead of decomposing a universal relation, we may prefer to synthesize a

$R^u \cap \cdots \cap R^u \cap \cdots \cap R^u$

since it involves all the attributes in the „universal“ defined by the relation $u \times \cdots \times \mathcal{T} \times u \mathcal{T}$ is called a universal relation

$\Pi_{\mathcal{T}}(u \mathcal{T})$

Relation:

– A tuple of relation $u \mathcal{T}$ is a dangling tuple if it is not in the

$\mathcal{T}$

– Let $\mathcal{T}$ be a set of relations.

Alternative Approaches to Database Design