CS422 Principles of Database Systems
Normalization

Chengyu Sun
California State University, Los Angeles

Schema Design

Problem Description → ER Design → ER Diagram → Relational Schema → ER to Relational Conversion

Transform Bad into Good

Good Schema?

Y

Good Relational Schema

Bad Schema

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>address</th>
<th>assignment</th>
<th>due</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>123 Main St.</td>
<td>HW1</td>
<td>2009-06-22</td>
<td>A-</td>
</tr>
<tr>
<td>1</td>
<td>John</td>
<td>123 Main St.</td>
<td>HW2</td>
<td>2009-07-10</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>Jane</td>
<td>456 State St.</td>
<td>HW1</td>
<td>2009-06-22</td>
<td>A</td>
</tr>
</tbody>
</table>

class_records

- Update anomaly
- Delete anomaly

Normalization

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>123 Main St.</td>
</tr>
<tr>
<td>2</td>
<td>Jane</td>
<td>456 State St.</td>
</tr>
</tbody>
</table>

students

<table>
<thead>
<tr>
<th>name</th>
<th>due</th>
</tr>
</thead>
<tbody>
<tr>
<td>HW1</td>
<td>2009-06-22</td>
</tr>
<tr>
<td>HW2</td>
<td>2009-07-10</td>
</tr>
</tbody>
</table>

assignments

<table>
<thead>
<tr>
<th>student</th>
<th>assignment</th>
<th>grades</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>HW1</td>
<td>A-</td>
</tr>
<tr>
<td>2</td>
<td>HW2</td>
<td>B</td>
</tr>
</tbody>
</table>

grades

Questions To Be Answered

- How do we decide whether a schema is bad?
- How do we decompose a table to turn a bad schema into a good one?

Functional Dependency (FD)

- A functional dependency on table R is the assertion that two records having the same values for attributes \{A_1, ..., A_n\} must also have the same value for attribute B
- \{A_1, ..., A_n\} \rightarrow B, or \{A_1, ..., A_n\} functionally determine B
About FD

A FD is an assertion based on assumptions about all possible data, not just the existing data.

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>(id) → (name)</th>
<th>✓</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>✓</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>Jane</td>
<td>×</td>
<td></td>
</tr>
</tbody>
</table>

FD with Multiple Attributes

\[
(A_1, A_2, A_3, ..., A_n) ightarrow B_1 \\
(A_1, A_2, A_3, ..., A_n) ightarrow B_2 \\
... \\
(A_1, A_2, A_3, ..., A_n) ightarrow B_m
\]

If \( \{A_1, A_2, A_3, ..., A_n\} \rightarrow \{B_1, B_2, B_3, ..., B_m\} \) is trivial then all \( B_m \) are in \( A_m \).

If \( \{A_1, A_2, A_3, ..., A_n\} \rightarrow \{B_1, B_2, B_3, ..., B_m\} \) is nontrivial then at least one \( B_m \) is not in \( A_m \).

From now on, when we talk about FD, we mean completely nontrivial FD unless otherwise noted.

Trivial Functional Dependency

FD: \( \{A_1, A_2, A_3, ..., A_n\} \rightarrow \{B_1, B_2, B_3, ..., B_m\} \)

- FD is trivial if all \( B_m \) are in \( A_m \)
- FD is nontrivial if at least one \( B_m \) is not in \( A_m \)
- FD is completely nontrivial if no \( B_m \) is in \( A_m \)

FD Example 1

- Musicians (id, name, address)
- Bands (id, name)
- Band_Members (band_id, musician_id)

FD Example 2

- Books (id, title)
- Authors (id, name)
- Book_Authors (book_id, author_id, author_order)

FD Example 3

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>address</th>
<th>assignment</th>
<th>due</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>123 Main St.</td>
<td>HW1</td>
<td>2009-06-22</td>
<td>A-</td>
</tr>
<tr>
<td>1</td>
<td>John</td>
<td>123 Main St.</td>
<td>HW2</td>
<td>2009-07-10</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>Jane</td>
<td>456 State St.</td>
<td>HW1</td>
<td>2009-06-22</td>
<td>A</td>
</tr>
</tbody>
</table>

Functional dependencies??
Key

A key of a table R if:
- A functionally determines all attributes of R
- No proper subset of A functionally determines all attributes of R

A Few Things about Keys

- A table may have multiple keys
- A key may consist of multiple attributes
- Superset of a key is called a super key
- The definition doesn’t say anything about uniqueness
- A key has to be **minimal**, but not necessarily **minimum**

Key Examples

- Musicians and bands
- Books and authors
- Class_records

Boyce-Codd Normal Form (BCNF)

- A table R is in BCNF if for every nontrivial FD A → B in R, A is a super key of R.

Or

*The key, the whole key, and nothing but the key, so help me Codd.*

BCNF or Not?

- Musicians and bands
- Books and authors
- Class_records

Determine If a Table is BCNF

- Step 1: identify all FDs
- Step 2: find all keys
- Step 3: check LHS of all non-trivial FDs and see if they are a superset of a key (i.e. a super key)
Decompose into BCNF

- Given table \( R \) with FD's \( F \)
- Look among \( F \) for a BCNF violation \( A \rightarrow B \)
- Compute \( A^+ \)
- Decompose \( R \) into:
  - \( R_1 = A^+ \)
  - \( R_2 = (R - A^+) \cup A \)
- Continue decomposition with \( R_1 \) and \( R_2 \) until all resulting tables are BCNF

Closure of Attributes \( A^+ \)

- Given
  - a set of attributes \( A \)
  - a set of functional dependencies \( S \)
- Closure of \( A \) under \( S, A^+ \), is the set of all possible attributes that are functionally determined by \( A \) based on the functional dependencies inferable from \( S \)

Simple Closure Example

- \( R: \{A, B, C\} \)
- \( S: \{A \rightarrow B, B \rightarrow C\} \)
- \( \{A\}^+ ?? \)
- \( \{B\}^+ ?? \)
- \( \{C\}^+ ?? \)

Armstrong’s Axioms

- Reflexivity
  - If \( B \subseteq A \), then \( A \rightarrow B \)
- Transitivity
  - If \( A \rightarrow B \) and \( B \rightarrow C \), then \( A \rightarrow C \)
- Augmentation
  - If \( A \rightarrow B \), then \( AC \rightarrow BC \) for any \( C \)

Two More FD Rules

- Union
  - If \( A \rightarrow B \) and \( A \rightarrow C \), then \( A \rightarrow BC \)
- Decomposition
  - If \( A \rightarrow BC \), then \( A \rightarrow B \) and \( A \rightarrow C \)

Computing \( A^+ \)

- Initialize \( A^+ = A \)
- Search in \( S \) for \( B \rightarrow C \) where
  - \( B \subseteq A^+ \)
  - \( C \in A^+ \)
- Add \( C \) to \( A^+ \)
- Repeat until nothing can be added to \( A^+ \)
Computing $A^+$ Example

- $R( A, B, C, D, E, F)$
- $S: AB \rightarrow C, BC \rightarrow AD, D \rightarrow E, CF \rightarrow B$
- $\{A,B\}^+$ ??
- Is $\{A,B\}$ a key ??
- How do we find out the key(s) from $R$ ??

Example: BCNF Decomposition

<table>
<thead>
<tr>
<th>id</th>
<th>name</th>
<th>address</th>
<th>assignment</th>
<th>due</th>
<th>grade</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>John</td>
<td>123 Main St.</td>
<td>HW1</td>
<td>2009-06-22</td>
<td>A</td>
</tr>
<tr>
<td>1</td>
<td>John</td>
<td>123 Main St.</td>
<td>HW2</td>
<td>2009-07-10</td>
<td>B</td>
</tr>
<tr>
<td>2</td>
<td>Jane</td>
<td>456 State St.</td>
<td>HW1</td>
<td>2009-06-22</td>
<td>A</td>
</tr>
</tbody>
</table>

Motivation for 3NF

- $\text{(street_address, city, zip}\rightarrow \text{zip})$
- $\text{(street_address, zip}\rightarrow \text{city})$

- We lose the FD $\text{(street_address, city)} \rightarrow \text{zip}$ after decomposition, or in other words, it becomes unenforceable.

An Unenforceable FD

Before decomposition:

<table>
<thead>
<tr>
<th>street</th>
<th>city</th>
<th>zip</th>
</tr>
</thead>
<tbody>
<tr>
<td>545 Tech Sq.</td>
<td>Cambridge</td>
<td>02138</td>
</tr>
<tr>
<td>545 Tech Sq.</td>
<td>Cambridge</td>
<td>02138</td>
</tr>
</tbody>
</table>

After decomposition:

<table>
<thead>
<tr>
<th>street</th>
<th>zip</th>
<th>city</th>
</tr>
</thead>
<tbody>
<tr>
<td>545 Tech Sq.</td>
<td>02138</td>
<td>Cambridge</td>
</tr>
<tr>
<td>545 Tech Sq.</td>
<td>02138</td>
<td>Cambridge</td>
</tr>
</tbody>
</table>

The same data error can no longer be detected.

Third Normal Form (3NF)

- A table $R$ is in 3NF if for every nontrivial FD $A \rightarrow B$ in $R$,
  - $A$ is a super key of $R$
  - or $B$ is part of a key of $R$

Schema Design