Relational Design Theory II

Normalization

Detecting Anomalies

<table>
<thead>
<tr>
<th>SID</th>
<th>Activity</th>
<th>Fee</th>
<th>Tax</th>
</tr>
</thead>
<tbody>
<tr>
<td>1001</td>
<td>Piano</td>
<td>$20</td>
<td>$2.00</td>
</tr>
<tr>
<td>1090</td>
<td>Swimming</td>
<td>$15</td>
<td>$1.50</td>
</tr>
<tr>
<td>1001</td>
<td>Swimming</td>
<td>$15</td>
<td>$1.50</td>
</tr>
</tbody>
</table>

• Why is this bad design?

• Can we capture this using FDs?

Normal Forms

• Requirements on relational schemas
  • Initiated by Codd (1NF, 2NF, 3NF)

<table>
<thead>
<tr>
<th>Normal Form</th>
<th>Requirements</th>
</tr>
</thead>
<tbody>
<tr>
<td>1NF (First NF)</td>
<td>no multivalued attributes</td>
</tr>
<tr>
<td>2NF (Second NF)</td>
<td>no partial dependencies</td>
</tr>
<tr>
<td>3NF (Third NF)</td>
<td>no bad transitive dependencies</td>
</tr>
<tr>
<td>BCNF (Boyce-Codd NF)</td>
<td>strengthening of 3NF</td>
</tr>
<tr>
<td>4NF (Fourth NF)</td>
<td>extends BCNF to multivalued dependencies</td>
</tr>
</tbody>
</table>

• there’s more …
BCNF

If $X \rightarrow Y$ is not trivial, then $X$ has to be a superkey.

Has to be true for all valid FDs $X \rightarrow Y$

Example:

Activity(SID, Activity, Fee, Tax)
SID, Activity $\rightarrow$ Fee, Tax
Activity $\rightarrow$ Fee
Fee $\rightarrow$ Tax

How to decompose?

BCNF Decomposition

What do we want?

- Relations are in BCNF
- We can reconstruct data in original relation
- Keep functional dependencies?

Note: Relations on two attributes are always BCNF.

BCNF-Normalization

Algorithm (BCNF Normalization)

Input: Relation R, FDs F
Output: BCNF-decomposition D of R

$D := \{R\}$
While $X \rightarrow Y$ holds in some $Q(\{A_1, ..., A_n\})$ in $D$, and $X \rightarrow Y$ not trivial, $X$ not a superkey of $Q$
add $Q_1(X^+ \setminus \{A_1, ..., A_n\})$ and $Q_2(X \cup \{A_1, ..., A_n\} - X^+)$
remove Q.
**BCNF-Example**

\(D := \{R\}\)

While \(X \rightarrow Y\) holds in some \(Q(A_1, \ldots, A_n)\) in \(D\), and

\(X \rightarrow Y\) not trivial, \(X\) not a superkey of \(Q\)

1. add \(Q', \{(A_1, \ldots, A_n)\}\) and
2. \(Q \leftarrow (Q \setminus \{(A_1, \ldots, A_n)\} \cup \{X\})\)

removes \(Q\)

Examples:

- \(R(A, B, C, D)\), FDs: \(A \rightarrow B, C \rightarrow D\)
- \(R(A, B, C, D)\), FDs: \(AC \rightarrow B, C \rightarrow D\)
- \(R(A,B,C,D,E)\), FDs: \(A \rightarrow BE, E \rightarrow D\)

---

**BCNF-Normalization Caveat**

Checking whether \(X \rightarrow Y\) holds in some \(Q\) in

\(D\) refers to \(F\), not just \(D\).

Example:

- \(R(A,B,C,D,E)\)
- FDs: \(A \rightarrow B\)
- \(BC \rightarrow D\) (implies \(AC \rightarrow D\))

- naive implementation of algorithm requires exponential time
- can be improved to polynomial time

(Toni Fischer, *Decomposition of a relation scheme into Boyce-Codd Normal Form*, ACM, SIGACT News, 1982)

---

**Testing for BCNF**

- single \(R\) with FDs \(F\)
- testing for BCNF can be done in polynomial time,
  it is sufficient to test \(F\)
- this is not true for decompositions, e.g.

\[\begin{align*}
R(A,B,C,D,E) & \quad \text{decompose into} \\
\text{FDs: } A \rightarrow B & \quad R_1(A,B) \\
BC \rightarrow D & \quad R_2(A,C,D,E)
\end{align*}\]
More BCNF-Examples

phone(Name, City, AreaCode, PhoneNumber, Extension)
R(A,B,C,D,E), FDs: A→B, C→D, AC→E

Lossless Join Property

D = [R₁, R₂, …, Rₙ] decomposition of R.

If R₁*R₂* … * Rₙ = R, then
D has the lossless join property.

BCNF decomposition has lossless join property.

Lossless Join Property

Test lossless join property for binary decomposition
Given: R, D= [R₁, R₂], FDs F
D is a lossless join decomposition of R, if
R = R₁ ∪ R₂, and either
R₁ ∩ R₂ → R₁ – R₂ in F⁺ or
R₁ ∩ R₂ → R₂ – R₁ in F⁺.

Example: Activity(SID, Activity, Fee)

• necessary?
• sufficient?
• implies correctness of BCNF algorithm
**Lossless Join Property**

**Algorithm (Chase Test)**

- **Input:** relation \( R(A_1, \ldots, A_n) \), FDs \( F \)
- **Decomposition:** \( D = \{ R_1, R_2, \ldots, R_m \} \)
- **Output:** Is \( D \) a lossless join decomposition of \( R \)?

\[
T := \text{table with columns } A_1, \ldots, A_n, \text{ rows } R_1, R_2, \ldots, R_m
\]

\[
T[i,j] := \begin{cases} 
    a_j & \text{if } A_i \text{ not in } R_j \\
    a_i & \text{if } A_i \text{ in } R_j 
\end{cases}
\]

Apply FDs in \( F \) to identify elements until
- there is a row \((a_1, \ldots, a_k)\): lossless join
- no more changes are possible: not lossless join

**Chase Test Examples**

- \( R(A,B,C) \), FDs: \( A \rightarrow B \), \( D = \{ P(A,B), Q(A,C) \} \)
- \( R(A,B,C) \), FDs: \( A \rightarrow B \), \( D = \{ P(B,C), Q(A,C) \} \)
- \( R(A,B,C,D) \), FDs: \( A \rightarrow B, C \rightarrow D \), \( D = \{ P(A,B), Q(B,C), T(C,D) \} \)

**Dependency Preservation**

- \( \text{banker}(\text{BranchName}, \text{CustomerName}, \text{BankerName}) \)
  - \( \text{BankerName} \rightarrow \text{BranchName} \)
  - \( \text{BranchName}, \text{CustomerName} \rightarrow \text{BankerName} \)

- \( R(A,B,C) \), FDs: \( A \rightarrow B, BC \rightarrow A \)
  - why not in BCNF? (Keys?)
  - what are possible BCNF decompositions?
  - what happens to dependencies?

Deciding whether a given relation has a dependency preserving BCNF decomposition is NP-complete

*Tsou, Fischer, Decomposition of a relation scheme into Boyce-Codd Normal Form, ACM, SIGACT News, 1982*
Prime Attributes

**prime attribute**: part of some key

**Examples:**
- \( R(A,B,C,D,E) \), \( AB \) is key, \( C \) is key, \( B \rightarrow D, D \rightarrow E \)
- \( A,B,C \) are prime,
  \( D,E \) are nonprime

\( R(A,B,C,D,E) \)
\( AC \rightarrow D, BD \rightarrow E, E \rightarrow AC \)

---

Prime Attributes

**prime attribute**: part of some key

- How many keys can there be on \( n \) attributes?
- How hard is it to find all keys? Algorithm?

---

Determining primality of an attribute is NP-complete.

3NF

If \( X \rightarrow Y \) is not trivial, then \( X \) has to be a superkey, or, all attributes in \( Y \cdot X \) are prime.

Has to be true for all valid FDs \( X \rightarrow Y \)

Violated in

- Book(Author, Title, PriceCategory, Price)
- Movie(Title, Year, MPAA, MinimumAge)

3NF-Examples

\[ R(A,B,C,D) \] with key \( A \) and
\[ B \rightarrow CD, \ C \rightarrow D, \ D \rightarrow C \]

Can we find a decomposition of these relations that contains the same information?

3NF-Normalization

**Algorithm (3NF Normalization):**

Input: Relation \( R \) with FDs \( F \)

Output: 3NF decomposition \( D \) of \( R \)

1. Compute canonical cover \( C \) of \( F \)
2. \( D = {} \)
3. For every \( X \rightarrow Y \) in \( C \) such that no \( S \) in \( D \) contains all of \( XY \), add new relation \( Q(XY) \) to \( D \)
4. If no relation in \( D \) contains a key of \( R \), then add new relation \( Q(X) \) on some key \( X \) of \( R \)
3NF-Examples

- R(A,B,C,D) with A key, B→CD, C→D, D→C
- R(A,B,C,D) with AB key, A→C, B→D
- R(A,B,C,D,E) with AB and AC keys, BC→D, C→E
- R(A,B,C,D,E) with AB key, A→E, BC→D, D→E
- R(A,B,C,D,E,F) with ABC key, A→E, AC→F, EF→G
- R(A,B,C,D,E,F) with A and BC keys, B→D, D→F
- R(A,B,C,D,E) with AB and CD keys, A→E, C→E

Find 3NF normalization
Results can depend on canonical cover, and order of execution

3NF Algorithm

- 3NF Normalization Algorithm is loss-less join (chase test)
- It is dependency preserving (obviously)
- The resulting relations are in 3NF (not trivial).

3NF vs BCNF: properties

- BCNF is stronger than 3NF
- BCNF and 3NF are loss-less join (no spurious tuples)
- 3NF preserves dependencies
- BCNF does not always preserve dependencies
3NF vs BCNF: algorithmics

- Normalization Algorithms:
  - naive algorithm for 3NF in polynomial time
  - naive algorithm for BCNF in exponential time, but can be done in polynomial time
- Recognition Algorithms:
  - BCNF is easy to recognize (polynomial time)
  - Recognizing 3NF is NP-complete

(Juri Fischer, The complexity of recognizing 3NF relation schemes, Information Processing Letters 14, 1982)