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>Description</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 …
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 $\mathcal{F}$
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 $Q(X \cap \{A_1, ..., A_n\})$ 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 \(Q\)

add \(Q/(X \cap (A_1, \ldots, A_n))\) and

\(Q_1(X \cup \{A_1, \ldots, A_n \setminus X\})\)

remove \(Q\)

**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 \(AC \rightarrow D\))

• naïve 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 dependencies in \(F\)
• this is not true for decompositions, e.g.

- \(R(A, B, C, D, E)\)
  - decompose into
- \(R_1(A, B)\)
- \(R_2(A, C, D, E)\)
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 $\mathcal{F}$
D is a lossless join decomposition of R, if and only if
R = R₁ U R₂, and either
R₁ ∩ R₂ $\rightarrow$ R₁ − R₂ holds in $\mathcal{F}$ or
R₁ ∩ R₂ $\rightarrow$ R₂ − R₁ holds in $\mathcal{F}$.

Example: Activity(SID, Activity, Fee)
• necessary?
• sufficient?
• implies correctness of BCNF algorithm
Lossless Join Property

Algorithm (Chase Test)
Input: relation R(A₁, …, Aₙ), FDs F
decomposition D = {R₁, R₂, …, Rₘ}
Output: Is D a lossless join decomposition of R?
T := table with columns A₁, …, Aₙ, rows R₁, R₂, …, Rₘ
T[i,j] :=
  aᵢ if Aᵢ not in Rᵢ
  aᵢ if Aᵢ in Rᵢ
Apply FDs in F to identify elements until
• there is a row (a₁, …, aₙ): lossless join
• no more changes are possible: not lossless join

Chase Test Examples

R(A,B,C), FDs: A → B,
D = {P(A,B), Q(A,C)}

R(A,B,C), FDs: A → B,
D = {P(B,C), Q(A,C)}

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

Dependency Preservation

banker(BranchName, CustomerName, BankerName)
BankerName → BranchName
BranchName, CustomerName → BankerName
R(A,B,C), FDs: A → B, BC → 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:**

\[ R(A,B,C,D,E) \]
\[ \text{AB is key, C is key, } B \rightarrow D, D \rightarrow E \]
\[ \text{A,B,C are prime, } D,E \text{ are nonprime} \]

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

---

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.

(Lucchesi, Osborne; Candidate keys for relations, J. Comput. System Sci. 17, 1978)
If \( X \rightarrow Y \) is not trivial, then \( X \) has to be a superkey, or all attributes in \( Y-X \) 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 \( C \) add \( Q(XY) \) to \( D \), unless
   a) some \( S \) in \( D \) already contains all of \( XY \): don’t add \( Q \)
   b) some \( S \) in \( D \) is contained in \( XY \): replace \( S \) with \( Q(XY) \)
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, $B \rightarrow CD$, $C \rightarrow D$, $D \rightarrow C$
- $R(A,B,C,D)$ with AB key, $A \rightarrow C$, $B \rightarrow D$
- $R(A,B,C,D,E)$ with AB and AC keys, $BC \rightarrow D$, $C \rightarrow E$
- $R(A,B,C,D,E)$ with AB key, $A \rightarrow E$, $BC \rightarrow D$, $D \rightarrow E$
- $R(A,B,C,D,E,F)$ with ABC key, $A \rightarrow E$, $AC \rightarrow F$, $EF \rightarrow G$
- $R(A,B,C,D,E,F)$ with A and BC keys, $B \rightarrow D$, $D \rightarrow F$
- $R(A,B,C,D,E)$ with AB and CD keys, $A \rightarrow E$, $C \rightarrow E$

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:**
  - naïve algorithm for 3NF in polynomial time
  - naïve 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