Introduction to Data Management
CSE 344

Lectures 18: BCNF
What makes good schemas?

**WHY SO MANY DATABASE TABLES???

I UPDATED A SCHEMA ONCE

IT SUCKED**
Review: Relation Decomposition

Break the relation into two:

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>SSN</th>
<th>PhoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-45-6789</td>
<td>206-555-1234</td>
</tr>
<tr>
<td>123-45-6789</td>
<td>206-555-6543</td>
</tr>
<tr>
<td>987-65-4321</td>
<td>908-555-2121</td>
</tr>
</tbody>
</table>

Anomalies have gone:

- No more repeated data
- Easy to move Fred to “Bellevue” (how ?)
- Easy to delete all Joe’s phone numbers (how ?)
Review: Functional Dependencies (FDs)

**Definition**

If two tuples agree on the attributes

\[ A_1, A_2, \ldots, A_n \]

then they must also agree on the attributes

\[ B_1, B_2, \ldots, B_m \]

Formally:

\[ A_1, A_2, \ldots, A_n \Rightarrow B_1, B_2, \ldots, B_m \]
Review: Functional Dependencies (FDs)

**Definition**  \(A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n\) holds in \(R\) if:

\[
\forall t, t' \in R, \\
(t.A_1 = t'.A_1 \land \ldots \land t.A_m = t'.A_m \rightarrow t.B_1 = t'.B_1 \land \ldots \land t.B_n = t'.B_n)
\]

<table>
<thead>
<tr>
<th>R</th>
<th>(A_1)</th>
<th>(\ldots)</th>
<th>(A_m)</th>
<th>(B_1)</th>
<th>(\ldots)</th>
<th>(B_n)</th>
</tr>
</thead>
<tbody>
<tr>
<td>t</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>t'</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

*if \(t, t'\) agree here then \(t, t'\) agree here*
Review: An Interesting Observation

If all these FDs are true:

- name $\rightarrow$ color
- category $\rightarrow$ department
- color, category $\rightarrow$ price

Then this FD also holds:

- name, category $\rightarrow$ price

If we find out from application domain that a relation satisfies some FDs, it doesn’t mean that we found all the FDs that it satisfies! There could be more FDs implied by the ones we have.
Review:
Closure of a set of Attributes

Given a set of attributes \( A_1, \ldots, A_n \)

The closure is the set of attributes \( B \), notated \( \{A_1, \ldots, A_n\}^+ \), s.t. \( A_1, \ldots, A_n \rightarrow B \)

Example:

1. name \( \rightarrow \) color
2. category \( \rightarrow \) department
3. color, category \( \rightarrow \) price

Closures:

\( \text{name}^+ = \{\text{name}, \text{color}\} \)
\( \{\text{name}, \text{category}\}^+ = \{\text{name}, \text{category}, \text{color}, \text{department}, \text{price}\} \)
\( \text{color}^+ = \{\text{color}\} \)
Closure Algorithm

\[ X = \{A_1, \ldots, A_n\} \]

Repeat until \( X \) doesn’t change do:

if \( B_1, \ldots, B_n \rightarrow C \) is a FD and \( B_1, \ldots, B_n \) are all in \( X \)

then add C to \( X \).

Example:

\[
\{ \text{name, category } \}^+ = \\
\{ \text{name, category, color, department, price } \}
\]

Hence: \( \text{name, category } \rightarrow \text{color, department, price } \)
Example

In class:

$R(A,B,C,D,E,F)$

\[
\begin{array}{c}
A, B \rightarrow C \\
A, D \rightarrow E \\
B \rightarrow D \\
A, F \rightarrow B
\end{array}
\]

Compute $\{A, B\}^+$ $X = \{A, B, \}$

Compute $\{A, F\}^+$ $X = \{A, F, \}$
Example

In class:

\[ R(A,B,C,D,E,F) \]

\[
\begin{array}{c}
A, B \rightarrow C \\
A, D \rightarrow E \\
B \rightarrow D \\
A, F \rightarrow B \\
\end{array}
\]

\[
\text{Compute } \{A, B\}^+ \quad X = \{A, B, C, D, E\}
\]

\[
\text{Compute } \{A, F\}^+ \quad X = \{A, F\}
\]
Example

In class:

\[ R(A,B,C,D,E,F) \]

\[
\begin{array}{c}
A, \ B \rightarrow \ C \\
A, \ D \rightarrow \ E \\
B \rightarrow \ D \\
A, \ F \rightarrow \ B
\end{array}
\]

Compute \{A, B\}^+ \quad X = \{A, B, C, D, E\}

Compute \{A, F\}^+ \quad X = \{A, F, B, C, D, E\}
Example

In class:

\[ R(A,B,C,D,E,F) \]

\[
A, B \rightarrow C \\
A, D \rightarrow E \\
B \rightarrow D \\
A, F \rightarrow B
\]

Compute \( \{A, B\}^+ \) \( X = \{A, B, C, D, E\} \)

Compute \( \{A, F\}^+ \) \( X = \{A, F, B, C, D, E\} \)

What is a key of \( R \)?
Keys

• A **superkey** is a set of attributes $A_1, \ldots, A_n$ s.t. for any other attribute $B$ in the same relation, we have $A_1, \ldots, A_n \rightarrow B$

• A **key** is a minimal superkey
  – A superkey and for which no subset is a superkey
Computing (Super)Keys

• For all sets $X$, compute $X^+$

• If $X^+ = [\text{all attributes}]$, then $X$ is a superkey

• Try reducing to the minimal $X$’s to get the key
Example

Product(name, price, category, color)

name, category $\rightarrow$ price
category $\rightarrow$ color

What is the key?
Example

Product(name, price, category, color)

(name, category) \rightarrow price
category \rightarrow color

What is the key?

\((\text{name}, \text{category})^+ = \{ \text{name}, \text{category}, \text{price}, \text{color}\}\)

Hence (name, category) is a key
Key or Keys?

Can we have more than one key?

Given \( R(A,B,C) \) define FD’s s.t. there are two or more distinct keys
Key or Keys?

Can we have more than one key?

Given R(A,B,C) define FD's s.t. there are two or more distinct keys

A → B
B → C
C → A

or

AB → C
BC → A

or

A → BC
B → AC

what are the keys here?
Eliminating Anomalies

Main idea:

• $X \rightarrow A$ is OK if $X$ is a (super)key

• $X \rightarrow A$ is not OK otherwise
  – Need to decompose the table, but how?

Boyce-Codd Normal Form
Boyce-Codd Normal Form

Dr. Raymond F. Boyce
Turing Awards in Data Management

Charles Bachman, 1973
*IDS and CODASYL*

Ted Codd, 1981
*Relational model*

Jim Gray, 1998
*Transaction processing*

Michael Stonebraker, 2014
*INGRES and Postgres*
Boyce-Codd Normal Form

There are no “bad” FDs:

**Definition.** A relation R is in BCNF if:
Whenever $X \rightarrow B$ is a non-trivial dependency, then $X$ is a superkey.

Equivalently:

**Definition.** A relation R is in BCNF if:

\[
\forall X \text{ in } X \rightarrow B, \quad \text{either } X^+ = X \text{ or } X^+ = [\text{all attributes}]
\]
BCNF Decomposition Algorithm

Normalize(R)

find X in X→B s.t.: X ≠ X⁺ and X⁺ ≠ [all attributes]

if (not found) then R is in BCNF

let Y = X⁺ - X; Z = [all attributes] - X⁺

decompose R into R₁(X ∪ Y) and R₂(X ∪ Z)

Normalize(R₁); Normalize(R₂);
Want $X$ in $X \rightarrow B$ s.t.: $X = X^+$ or $X^+ = \{\text{all attributes}\}$

**Example**

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>908-555-1234</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

**SSN $\rightarrow$ Name, City**

The only key is: $\{\text{SSN, PhoneNumber}\}$

Hence $\text{SSN $\rightarrow$ Name, City}$ is a “bad” dependency

In other words:

$\text{SSN}^+ = \text{SSN, Name, City}$ and is neither $\text{SSN}$ nor $\text{All Attributes}$
Example BCNF Decomposition

Want X in $X \rightarrow B$ s.t.: $X = X^+$ or $X^+ = [\text{all attributes}]$

Let's check anomalies:
- Redundancy?
- Update?
- Delete?

### Example Table

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987-65-4321</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

### Phone Numbers Table

<table>
<thead>
<tr>
<th>SSN</th>
<th>PhoneNumber</th>
</tr>
</thead>
<tbody>
<tr>
<td>123-45-6789</td>
<td>206-555-1234</td>
</tr>
<tr>
<td>123-45-6789</td>
<td>206-555-6543</td>
</tr>
<tr>
<td>987-65-4321</td>
<td>908-555-2121</td>
</tr>
<tr>
<td>987-65-4321</td>
<td>908-555-1234</td>
</tr>
</tbody>
</table>
Find $X$ in $X \rightarrow B$ s.t.: $X \neq X^+$ and $X^+ \neq [\text{all attributes}]$

**Example BCNF Decomposition**

Person($\text{name, SSN, age, hairColor, phoneNumber}$)

- SSN $\rightarrow$ name, age
- age $\rightarrow$ hairColor
Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age

age → hairColor

Iteration 1: Person: SSN+ = SSN, name, age, hairColor

Decompose into: P(SSN, name, age, hairColor)

Phone(SSN, phoneNumber)
Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN $\rightarrow$ name, age
age $\rightarrow$ hairColor

Iteration 1: Person: $SSN^+ = SSN, name, age, hairColor$
Decompose into: $P(SSN, name, age, hairColor)$
Phone(SSN, phoneNumber)

Iteration 2: $P$: age$^+ = age, hairColor$
Decompose: People(SSN, name, age)
Hair(age, hairColor)
Phone(SSN, phoneNumber)

What are the keys?
Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age
age → hairColor

Iteration 1: Person: SSN+ = SSN, name, age, hairColor
Decompose into: P(SSN, name, age, hairColor)
              Phone(SSN, phoneNumber)

Iteration 2: P: age+ = age, hairColor
Decompose: People(SSN, name, age)
           Hair(age, hairColor)
           Phone(SSN, phoneNumber)

Find X in X→B s.t.: X ≠X+ and X+ ≠ [all attributes]
Example: BCNF

R(A,B,C,D)

A → B
B → C
Recall: find $X$ s.t. $X \neq X^+ \neq [\text{all-attrs}]$
Recall: find X s.t. 
\[ X \neq X^+ \neq [\text{all-attrs}] \]

Example: BCNF

\[ R(A,B,C,D) \]

\[ A^+ = ABC \neq ABCD \]
Example: BCNF

Recall: find $X$ s.t. $X \neq X^+ \neq \text{[all-attrs]}$

$R(A,B,C,D)$

$A^+ = ABC \neq ABCD$

$R_1(A,B,C)$

$R_2(A,D)$
Example: BCNF

Recall: find $X$ s.t. $X \neq X^+ \neq \{\text{all-attrs}\}$

$R(A,B,C,D)$

$A^+ = ABC \neq ABCD$

$R_1(A,B,C)$

$B^+ = BC \neq ABC$

$R_2(A,D)$

$A \rightarrow B$

$B \rightarrow C$
Example: BCNF

R(A,B,C,D)

A → B
B → C

Recall: find X s.t. X ≠ X⁺ ≠ [all-attrs]

R(A,B,C,D)
A⁺ = ABC ≠ ABCD

What happens if in R we first pick B⁺? Or AB⁺?
Decompositions in General

$R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p)$

$S_1(A_1, \ldots, A_n, B_1, \ldots, B_m)$

$S_2(A_1, \ldots, A_n, C_1, \ldots, C_p)$

$S_1 = \text{projection of } R \text{ on } A_1, \ldots, A_n, B_1, \ldots, B_m$

$S_2 = \text{projection of } R \text{ on } A_1, \ldots, A_n, C_1, \ldots, C_p$
# Lossless Decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>
Lossy Decomposition

What is lossy here?

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Camera</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>
Lossy Decomposition

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Gadget</td>
</tr>
<tr>
<td>OneClick</td>
<td>24.99</td>
<td>Camera</td>
</tr>
<tr>
<td>Gizmo</td>
<td>19.99</td>
<td>Camera</td>
</tr>
</tbody>
</table>
Decomposition in General

**Fact:** If \( A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m \) then the decomposition is lossless.

It follows that every BCNF decomposition is lossless.
Schema Refinements
= Normal Forms

• 1st Normal Form = all tables are flat
• 2nd Normal Form = obsolete
• Boyce Codd Normal Form = no bad FDs
• 3rd and 4th Normal Form = see book
  – BCNF is lossless but can cause loss of ability to check some FDs (see book 3.4.4)
  – 3NF fixes that (is lossless and dependency-preserving), but some tables might not be in BCNF – i.e., they may have redundancy anomalies