Announcements

• HW6 and WQ6 are out
  – Due on Monday, 11/21

• Midterms graded
  – Submit regrade requests online

• This week
  – Conceptual (i.e., schema) design
What makes good schemas?
Relational Schema Design

Anomalies:
• Redundancy = repeat data
• Update anomalies = what if Fred moves to “Bellevue”?
• Deletion anomalies = what if Joe deletes his phone number?

<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>
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>

Anomalies have gone:
- No more repeated data
- Easy to move Fred to “Bellevue” (how?)
- Easy to delete all Joe’s phone numbers (how?)
Review: Keys

• A **superkey** of R is a set of attributes $A_1, \ldots, A_n$ s.t. for any other attribute $B$ in R, 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
Review: 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
Review: 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
Edgar Frank “Ted” Codd

"A Relational Model of Data for Large Shared Data Banks"
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:

$\forall X$, either $X^+ = X$ or $X^+ = [\text{all attributes}]$
BCNF Decomposition Algorithm

\[
\text{Normalize}(R) \\
\text{find } X \text{ s.t.: } X \neq X^+ \text{ and } X^+ \neq [\text{all attributes}] \\
\text{if (not found) then } R \text{ is in BCNF} \\
\text{let } Y = X^+ - X; \quad Z = [\text{all attributes}] - X^+ \\
\text{decompose } R \text{ into } R_1(X \cup Y) \text{ and } R_2(X \cup Z) \\
\text{Normalize}(R_1); \quad \text{Normalize}(R_2);
\]
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: \{SSN, PhoneNumber\}
Hence \(\text{SSN} \rightarrow \text{Name, City}\) 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

<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>

### Let’s check anomalies:
- Redundancy?
- Update?
- Delete?

<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 s.t.: $X \neq X^+$ and $X^+ \neq \{\text{all attributes}\}$

**Example BCNF Decomposition**

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

- SSN → name, age
- age → hairColor
Find X s.t.: $X \neq X^+$ and $X^+ \neq$ [all attributes]

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)
Example BCNF Decomposition

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

\[ \text{SSN} \rightarrow \text{name, age} \]
\[ \text{age} \rightarrow \text{hairColor} \]

\begin{itemize}
  \item \textbf{Iteration 1: Person:} \quad \text{SSN}^+ = \text{SSN, name, age, hairColor} \\
  \quad \text{Decompose into: } P(\text{SSN, name, age, hairColor}) \\
  \quad \quad \text{Phone(SSN, phoneNumber)}
\end{itemize}

\begin{itemize}
  \item \textbf{Iteration 2: P:} \quad \text{age}^+ = \text{age, hairColor} \\
  \quad \text{Decompose: } \text{People}(\text{SSN, name, age}) \\
  \quad \quad \text{Hair(age, hairColor)} \\
  \quad \quad \text{Phone(SSN, phoneNumber)}
\end{itemize}
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)
Example: BCNF
R(A,B,C,D)

Example: BCNF

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

R(A,B,C,D)

A → B
B → C
Example: BCNF

R(A,B,C,D)

$A^+ = ABC \neq ABCD$

A → B
B → C
Example: BCNF

R(A, B, C, D)

A\rightarrow B
B\rightarrow C

R(A,B,C,D)

A^+ = ABC \neq ABCD

R_1(A,B,C)

R_2(A,D)
Example: BCNF

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

\[ A^+ = ABC \neq ABCD \]

\[ R_1(A,B,C) \]
\[ B^+ = BC \neq ABC \]

\[ R_2(A,D) \]

A \rightarrow B
B \rightarrow C
Example: BCNF

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

$A^+ = ABC \neq ABCD$

$R_1(A,B,C)$

$B^+ = BC \neq ABC$

$R_{11}(B,C)$

$R_{12}(A,B)$

$R_2(A,D)$

What are the keys?

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>

- **Name**
  - Gizmo
  - OneClick
  - Gizmo

- **Price**
  - 19.99
  - 24.99
  - 19.99

- **Category**
  - Gadget
  - Camera
**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>
</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>

CSE 344 - Fall 2016 27
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>

<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>
Decomposition in General

Let:

\[ S_1 = \text{projection of } R \text{ on } A_1, \ldots, A_n, B_1, \ldots, B_m \]
\[ S_2 = \text{projection of } R \text{ on } A_1, \ldots, A_n, C_1, \ldots, C_p \]

The decomposition is called **lossless** if \( R = S_1 \bowtie S_2 \)

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
The Chase Test for Lossless Join

$R(A,B,C,D) = S_1(A,D) \bowtie S_2(A,C) \bowtie S_3(B,C,D)$

$R$ satisfies: $A \rightarrow B, B \rightarrow C, CD \rightarrow A$

$S_1 = \Pi_{AD}(R), S_2 = \Pi_{AC}(R), S_3 = \Pi_{BCD}(R)$,

hence $R \subseteq S_1 \bowtie S_2 \bowtie S_3$

Need to check: $R \supseteq S_1 \bowtie S_2 \bowtie S_3$
Example from textbook Ch. 3.4.2

The Chase Test for Lossless Join

\[ R(A,B,C,D) = S_1(A,D) \bowtie S_2(A,C) \bowtie S_3(B,C,D) \]

\( R \) satisfies: \( A \rightarrow B \), \( B \rightarrow C \), \( CD \rightarrow A \)

\( S_1 = \Pi_{AD}(R), \; S_2 = \Pi_{AC}(R), \; S_3 = \Pi_{BCD}(R), \)

hence \( R \subseteq S_1 \bowtie S_2 \bowtie S_3 \)

Need to check: \( R \supseteq S_1 \bowtie S_2 \bowtie S_3 \)

Suppose \((a,b,c,d) \in S_1 \bowtie S_2 \bowtie S_3\) Is it also in \( R \)?

\( R \) must contain the following tuples:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
</tr>
</tbody>
</table>

Why?

\((a,d) \in S_1 = \Pi_{AD}(R)\)
The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)

R satisfies: A \rightarrow B, B \rightarrow C, CD \rightarrow A

S1 = \Pi_{AD}(R), S2 = \Pi_{AC}(R), S3 = \Pi_{BCD}(R),
hence R \subseteq S1 \bowtie S2 \bowtie S3

Need to check: R \supseteq S1 \bowtie S2 \bowtie S3

Suppose (a, b, c, d) \in S1 \bowtie S2 \bowtie S3 Is it also in R?

R must contain the following tuples:

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
<td>c</td>
<td>d2</td>
</tr>
</tbody>
</table>

(a,d) \in S1 = \Pi_{AD}(R)

(a,c) \in S2 = \Pi_{AC}(R)
The Chase Test for Lossless Join

Example from textbook Ch. 3.4.2

\[ R(A, B, C, D) = S1(A, D) \bowtie S2(A, C) \bowtie S3(B, C, D) \]

R satisfies: \( A \rightarrow B, \ B \rightarrow C, \ CD \rightarrow A \)

\( S1 = \Pi_{AD}(R), \ S2 = \Pi_{AC}(R), \ S3 = \Pi_{BCD}(R), \)

hence \( R \subseteq S1 \bowtie S2 \bowtie S3 \)

Need to check: \( R \supseteq S1 \bowtie S2 \bowtie S3 \)

Suppose \((a, b, c, d) \in S1 \bowtie S2 \bowtie S3\) Is it also in R?

R must contain the following tuples:

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>a3</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

Why ?

\((a, d) \in S1 = \Pi_{AD}(R)\)

\((a, c) \in S2 = \Pi_{AC}(R)\)

\((b, c, d) \in S3 = \Pi_{BCD}(R)\)
The Chase Test for Lossless Join

R(A,B,C,D) = S1(A,D) ⋈ S2(A,C) ⋈ S3(B,C,D)
R satisfies: A → B, B → C, CD → A

S1 = Π_{AD}(R), S2 = Π_{AC}(R), S3 = Π_{BCD}(R),
hence R ⊆ S1 ⋈ S2 ⋈ S3

Need to check: R ⊇ S1 ⋈ S2 ⋈ S3
Suppose (a,b,c,d) ∈ S1 ⋈ S2 ⋈ S3 Is it also in R?
R must contain the following tuples:

“Chase” them (apply FDs):

<table>
<thead>
<tr>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
<td>c</td>
<td>d2</td>
</tr>
<tr>
<td>a3</td>
<td>b</td>
<td>c</td>
<td>d</td>
</tr>
</tbody>
</table>

Why?

(a,d) ∈ S1 = Π_{AD}(R)
(a,c) ∈ S2 = Π_{AC}(R)
(b,c,d) ∈ S3 = Π_{BCD}(R)
The Chase Test for Lossless Join

Example from textbook Ch. 3.4.2

\[ \text{R}(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D) \]
R satisfies: \( A \rightarrow B, \ B \rightarrow C, \ CD \rightarrow A \)

\[ S1 = \Pi_{AD}(R), \ S2 = \Pi_{AC}(R), \ S3 = \Pi_{BCD}(R), \]
hence \( R \subseteq S1 \bowtie S2 \bowtie S3 \)

Need to check: \( R \supseteq S1 \bowtie S2 \bowtie S3 \)

Suppose \((a,b,c,d) \in S1 \bowtie S2 \bowtie S3\) Is it also in \( R \)?

R must contain the following tuples:

```
A B C D
a b1 c1 d
a b2 c d2
a3 b c d
```

“Chase” them (apply FDs):

\( A \rightarrow B \)

\[
\begin{array}{cccc}
A & B & C & D \\
\hline
a & b1 & c1 & d \\
\hline
a & b1 & c & d2 \\
\hline
a3 & b & c & d \\
\end{array}
\]

\( B \rightarrow C \)

\[
\begin{array}{cccc}
A & B & C & D \\
\hline
a & b1 & c1 & d \\
\hline
a & b1 & c & d2 \\
\hline
a3 & b & c & d \\
\end{array}
\]

Why?

\( (a,d) \in S1 = \Pi_{AD}(R) \)

\( (a,c) \in S2 = \Pi_{AC}(R) \)

\( (b,c,d) \in S3 = \Pi_{BCD}(R) \)
The Chase Test for Lossless Join

Example from textbook Ch. 3.4.2

\[
R(A,B,C,D) = S1(A,D) \bowtie S2(A,C) \bowtie S3(B,C,D)
\]

\( R \) satisfies: \( A \rightarrow B, \ B \rightarrow C, \ CD \rightarrow A \)

\( S1 = \Pi_{AD}(R), \ S2 = \Pi_{AC}(R), \ S3 = \Pi_{BCD}(R), \)

hence \( R \subseteq S1 \bowtie S2 \bowtie S3 \)

Need to check: \( R \supseteq S1 \bowtie S2 \bowtie S3 \)

Suppose \((a,b,c,d) \in S1 \bowtie S2 \bowtie S3\) Is it also in \( R \)?

\( R \) must contain the following tuples:

"Chase" them (apply FDs):

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b1</td>
<td>c</td>
<td>d2</td>
<td></td>
</tr>
<tr>
<td>a3</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

Why?

<table>
<thead>
<tr>
<th></th>
<th>A</th>
<th>B</th>
<th>C</th>
<th>D</th>
</tr>
</thead>
<tbody>
<tr>
<td>a</td>
<td>b1</td>
<td>c1</td>
<td>d</td>
<td></td>
</tr>
<tr>
<td>a</td>
<td>b2</td>
<td>c</td>
<td>d2</td>
<td></td>
</tr>
<tr>
<td>a3</td>
<td>b</td>
<td>c</td>
<td>d</td>
<td></td>
</tr>
</tbody>
</table>

\((a,d) \in S1 = \Pi_{AD}(R)\)
\((a,c) \in S2 = \Pi_{AC}(R)\)
\((b,c,d) \in S3 = \Pi_{BCD}(R)\)

Hence \( R \) contains \((a,b,c,d)\)
Schema Refinements = Normal Forms

- 1st Normal Form = all tables are flat
- 2nd Normal Form = obsolete
- Boyce Codd Normal Form = no bad FDs
- 3rd 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