Announcements

- Midterm is posted and grades are up
  - Have until 8pm tonight for regrade requests
- WQ6 - 11pm July 31 (Monday)
- HW6 is due Aug 1 (Tuesday)
  - Database design
- Will drop HW8 - not enough time to get to Spark lecture before it is due.

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>988-555-2121</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: Functional Dependencies (FDs)

**Definition**

\[ A_1, ..., A_m \rightarrow B_1, ..., B_n \text{ holds in } R \text{ if:} \]

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

Example:

- \( \{\text{name}, \text{category}\} \rightarrow \{\text{color}, \text{department}, \text{price}\} \)
- \( \{\text{name}, \text{category}\} \rightarrow \{\text{color}, \text{department}, \text{price}\} \)

Hence:

- A superkey is a set of attributes \( A_1, ..., A_n \) s.t. for any other attribute \( B \) in the same relation, we have \( A_1, ..., A_n \rightarrow B \)
- A key is a minimal superkey – A superkey and for which no subset is a superkey

Keys

Closure Algorithm

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

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

- if \( B_1, ..., B_n \rightarrow C \) is a FD and \( B_1, ..., B_n \) are all in \( X \)
  - then add \( C \) to \( X \).

Example:

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

\{name, category\} \( \rightarrow \) color, department, price

- A superkey is a set of attributes \( A_1, ..., A_n \) s.t. for any other attribute \( B \) in the same relation, we have \( A_1, ..., A_n \rightarrow B \)
- 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?

(name, category)* = \{ name, category, price, color \}
Hence (name, category) is a key

Example

Product(name, price, category, color)

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

What is the key?

(name, category)* = \{ name, category, price, color \}
Hence (name, category) is a key

Example

Product(name, price, category, color)

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

What is the key?

(name, category)* = \{ name, category, price, color \}
Hence (name, category) is a key

How To Eliminate 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

There are no "bad" FĐs:

**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:
For all $X$ in $X \rightarrow B$:
- either $X^* = X$,
- or $X^* = \{\text{all attributes}\}$

BCNF Decomposition Algorithm

**Normalize**($R$)

- find $X$ in $X \rightarrow B$ s.t.: $X \neq X^*$ and $X^* \neq \{\text{all attributes}\}$
- if (not found) then $R$ is in BCNF
- **let** $Y = X^* - X$; $Z = \{\text{all attributes}\} - X^*$
- decompose $R$ into $R1(X \cup Y)$ and $R2(X \cup Z)$
- **normalize**($R1$); **normalize**($R2$);
The only key is: 
\( \{\text{SSN, PhoneNumber}\} \)

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 \( \text{All Attributes} \).

<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>

Want \( X \rightarrow B \) s.t. \( X = X' \) or \( X' = \{\text{all attributes}\} \):

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

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

Find \( X \rightarrow B \) s.t. \( X \neq X' \) and \( X' \neq \{\text{all attributes}\} \):

<table>
<thead>
<tr>
<th>Person(name, SSN, age, hairColor, phoneNumber)</th>
</tr>
</thead>
<tbody>
<tr>
<td>SSN \rightarrow name, age, hairColor</td>
</tr>
</tbody>
</table>

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

Iteration 2: \( P \): \( \text{age} = \text{age, hairColor} \)
Decompose: \( \text{People}(\text{SSN, name, age}) \)
\( \text{Hair}(\text{age, hairColor}) \)
\( \text{Phone}(\text{SSN, phoneNumber}) \)

Note the keys!
Example: BCNF

R(A,B,C,D)

Recall: find X s.t.
X ≠ X' ≠ [all-attrs]
R(A,B,C,D)
A* = ABC ≠ ABCD

R1(A,B,C)
R2(A,D)

What are the keys?

What happens if in R we first pick B*? Or AB*?

Decompositions in General

R(A1, ..., An, B1, ..., Bm, C1, ..., Cp)
S1(A1, ..., An, B1, ..., Bm)
S2(A1, ..., An, C1, ..., Cp)

S1 = projection of R on A1, ..., An, B1, ..., Bm
S2 = projection of R on A1, ..., An, C1, ..., Cp

Not all decompositions are good - some are lossy
A Lossless Decomposition

\[
\begin{array}{|c|c|c|}
\hline
\text{Name} & \text{Price} & \text{Category} \\
\hline
\text{Gizmo} & 19.99 & \text{Gadget} \\
\text{OneClick} & 24.99 & \text{Camera} \\
\hline
\end{array}
\]

A Lossy Decomposition

\[
\begin{array}{|c|c|c|}
\hline
\text{Name} & \text{Price} & \text{Category} \\
\hline
\text{Gizmo} & 19.99 & \text{Gadget} \\
\text{OneClick} & 24.99 & \text{Camera} \\
\hline
\end{array}
\]

Lossy Decomposition

\[
\begin{array}{|c|c|c|}
\hline
\text{Name} & \text{Price} & \text{Category} \\
\hline
\text{Gizmo} & 19.99 & \text{Gadget} \\
\text{OneClick} & 24.99 & \text{Camera} \\
\hline
\end{array}
\]

Decomposition in General

\[
R(A_1, \ldots, A_n, B_1, \ldots, B_m, C_1, \ldots, C_p) \\
S_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \quad S_2(A_1, \ldots, A_n, C_1, \ldots, C_p)
\]

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.

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

Normalization IRL

- It's good to know about BCNF (and maybe 3NF).
- Should every table always be in BCNF?

\[
\text{What are the FDs:} \\
C \rightarrow A_2, A_5, A_6, \ldots, A_9, B_1, S \\
A_5 \rightarrow A_7, A_8, \ldots, A_{10}, S \\
A_{10} \rightarrow A_7, A_8, \ldots, A_{10}, S \\
A_{10} \rightarrow A_7, A_8, \ldots, A_{10}, S
\]
Normalization IRL

R( Item, OrderNo, Customer, Address, Zip Code, State)

What are the FDs:

OrderNo -> Customer
Customer -> Address, Zip Code, State
Zip Code -> State

What is BCNF:

R(Item, OrderNo, Customer, Address, Zip Code, State)

OrderNo = Customer, Address, Zip, Code, State

R1(Item, OrderNo)
R2(OrderNo, Customer, Address, Zip Code, State)
R3(OrderNo, CustomerId)
R4(CustomerId, Name, Address, Zip Code)
R5(Zip Code, State)

Are We Done?

Normalization IRL

R( Item, OrderNo, Customer, Address, Zip Code, State)

What is BCNF:

R(Item, OrderNo, Customer, Address, Zip Code, State)

OrderNo* = Customer, Address, Zip, Code, State

R1(Item, OrderNo)
R2(OrderNo, Customer, Address, Zip Code, State)
R3(OrderNo, CustomerId)
R4(CustomerId, Name, Address, Zip Code)
R5(Zip Code, State)

Is this a good idea?