CSE 344

MAY 16TH – NORMALIZATION
ADMINISTRIVIA

• HW6 Due Tonight
  • Prioritize local runs
• OQ6 Out Today
• HW7 Out Today
  • E/R + Normalization
• Exams
  • In my office; Regrades through me
DATABASE DESIGN PROCESS

Conceptual Model:

Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details
Physical Schema
One person may have multiple phones, but lives in only one city.

Primary key is thus (SSN, PhoneNumber)

What is the problem with this schema?
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?)
How do we do this systematically?

Start with some relational schema.

Find out its functional dependencies (FDs).

Use FDs to normalize the relational schema.
Functional Dependencies (FDS)

**Definition**

If two tuples agree on the attributes

\[ A_1, A_2, \ldots, A_n, B_1, B_2, \ldots, B_m \]

then they must also agree on the attributes

Formally:

\[ A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m \]

\( A_1 \ldots A_n \text{ determines } B_1 \ldots B_m \)
**FUNCTIONAL DEPENDENCIES (FDS)**

**Definition**  
$A_1, ..., A_m \rightarrow B_1, ..., B_n$ holds in $R$ if:

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

$= t'.B_n$

<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
EXAMPLE

An FD holds, or does not hold on an instance:

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>Phone</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E3542</td>
<td>Mike</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

EmpID → Name, Phone, Position
Position → Phone
but not Phone → Position
<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>Phone</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E3542</td>
<td>Mike</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

**Position → Phone**
# Example

<table>
<thead>
<tr>
<th>EmpID</th>
<th>Name</th>
<th>Phone</th>
<th>Position</th>
</tr>
</thead>
<tbody>
<tr>
<td>E0045</td>
<td>Smith</td>
<td>1234</td>
<td>Clerk</td>
</tr>
<tr>
<td>E3542</td>
<td>Mike</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E1111</td>
<td>Smith</td>
<td>9876</td>
<td>Salesrep</td>
</tr>
<tr>
<td>E9999</td>
<td>Mary</td>
<td>1234</td>
<td>Lawyer</td>
</tr>
</tbody>
</table>

But not Phone $\rightarrow$ Position
Do all the FDs hold on this instance?

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>color</th>
<th>department</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
<td>Green</td>
<td>Toys</td>
<td>49</td>
</tr>
<tr>
<td>Tweaker</td>
<td>Gadget</td>
<td>Green</td>
<td>Toys</td>
<td>99</td>
</tr>
</tbody>
</table>
## EXAMPLE

<table>
<thead>
<tr>
<th>name</th>
<th>category</th>
<th>color</th>
<th>department</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
<td>Green</td>
<td>Toys</td>
<td>49</td>
</tr>
<tr>
<td>Tweaker</td>
<td>Gadget</td>
<td>Green</td>
<td>Toys</td>
<td>49</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Stationary</td>
<td>Green</td>
<td>Office-supp.</td>
<td>59</td>
</tr>
</tbody>
</table>

What about this one?
BUZZWORDS

FD holds or does not hold on an instance

If we can be sure that every instance of $R$ will be one in which a given FD is true, then we say that $R$ satisfies the FD

If we say that $R$ satisfies an FD, we are stating a constraint on $R$
WHY BOTHER WITH FDS?

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>
AN INTERESTING OBSERVATION

If all these FDs are true:

- name → color
- category → department
- color, category → price

Then this FD also holds:

- name, category → 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.
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:

\[
\begin{align*}
\text{name}^+ & = \{\text{name, color}\} \\
\{\text{name, category}\}^+ & = \{\text{name, category, color, department, price}\} \\
\text{color}^+ & = \{\text{color}\}
\end{align*}
\]
CLOSURE ALGORITHM

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

Repeat until \(X\) doesn’t change do:

\[ \text{if } B_1, \ldots, B_n \rightarrow C \text{ is a FD and } B_1, \ldots, B_n \text{ are all in } X \]
\[ \text{then add } C \text{ to } X. \]

Example:

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

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

Hence:

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

In class:

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

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

Compute \( \{A, F\}^+ \) \( X = \{A, F, \} \)
EXAMPLE

In class:

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

\[
\begin{align*}
A, B & \rightarrow C \\
A, D & \rightarrow E \\
B & \rightarrow D \\
A, F & \rightarrow B
\end{align*}
\]

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

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

In class:

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

\[
\begin{align*}
A, B & \rightarrow C \\
A, D & \rightarrow E \\
B & \rightarrow D \\
A, F & \rightarrow B
\end{align*}
\]

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

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

What is the key of \( R \)?
PRACTICE AT HOME

Find all FD’s implied by:

A, B → C
A, D → B
B → D
PRACTICE AT HOME

Find all FD’s implied by:

\[
\begin{align*}
A, B & \rightarrow C \\
A, D & \rightarrow B \\
B & \rightarrow D
\end{align*}
\]

Step 1: Compute \( X^+ \), for every \( X \):

\[
\begin{align*}
AB^+ &= ABCD, & AC^+ &= AC, & AD^+ &= ABCD, & BC^+ &= BCD, & BD^+ &= BD, & CD^+ &= CD \\
ABC^+ &= ABD^+ = ACD^+ = ABCD \text{ (no need to compute— why ?)} \\
BCD^+ &= BCD, & ABCD^+ &= ABCD
\end{align*}
\]

Step 2: Enumerate all FD’s \( X \rightarrow Y \), s.t. \( Y \subseteq X^+ \) and \( X \cap Y = \emptyset \):

\[
\begin{align*}
AB & \rightarrow CD, & AD & \rightarrow BC, & ABC & \rightarrow D, & ABD & \rightarrow C, & ACD & \rightarrow B
\end{align*}
\]
A superkey is a set of attributes $A_1, ..., A_n$ s.t. for any other attribute $B$, 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 → price
category → color

What is the key?
EXAMPLE

Product(name, price, category, color)

name, category → price
category → color

What is the key?

(name, category) + = { name, category, price, color }

Hence (name, category) is a key
Can we have more than one key?

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

- $A \rightarrow B$
- $B \rightarrow C$
- $C \rightarrow A$

or

- $AB \rightarrow C$
- $BC \rightarrow A$

or

- $A \rightarrow BC$
- $B \rightarrow AC$

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
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**

Normalize(R)

find X 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 \( R_1(X \cup Y) \) and \( R_2(X \cup Z) \)

Normalize(R1); Normalize(R2);
The only key is: \{SSN, PhoneNumber\}
Hence \(SSN \rightarrow \text{Name, City}\) is a “bad” dependency

In other words:
\(SSN^+ = SSN, \text{Name, City}\) and is neither \(SSN\) nor 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>
# 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?
Find $X$ s.t.: $X \neq X^+$ and $X^+ \neq \{\text{all attributes}\}$

**EXAMPLE BCNF DECOMPOSITION**

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

SSN $\rightarrow$ name, age

age $\rightarrow$ hairColor
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

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 s.t.: X ≠X+ and X+ ≠ [all attributes]
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 s.t.: X ≠X+ and X+ ≠ [all attributes]
EXAMPLE: BCNF

\[ R(A, B, C, D) \]
EXAMPLE: BCNF

R(A,B,C,D)

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

\[ R(A, B, C, D) \]
\[ A^+ = ABC \neq ABCD \]
EXAMPLE: BCNF

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

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

\[ R_1(A,B,C) \]

\[ R_2(A,D) \]
EXAMPLE: BCNF

R(A, B, C, D)
A⁺ = ABC ≠ ABCD

R₁(A, B, C)
B⁺ = BC ≠ ABC

R₂(A, D)

A → B
B → C
EXAMPLE: BCNF

R(A,B,C,D)
A+ = ABC ≠ ABCD

R1(A,B,C)
B+ = BC ≠ ABC

R11(B,C)

R12(A,B)

R2(A,D)

What are the keys?

What happens if in R we first pick B+? Or AB+?
DECOMPOSITIONS IN GENERAL

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

\[ S_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \]

\[ S_2(A_1, \ldots, A_n, C_1, \ldots, C_p) \]

\( S_1 = \) projection of \( R \) on \( A_1, \ldots, A_n, B_1, \ldots, B_m \)

\( S_2 = \) projection of \( R \) on \( A_1, \ldots, A_n, C_1, \ldots, C_p \)