CSE 344

AUGUST 3RD
NORMALIZATION
ADMINISTRIVIA

• WQ6 due Monday
  • DB design

• HW7 due next Wednesday
  • DB design
  • normalization
DATABASE DESIGN PROCESS

Conceptual Model:

Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

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

Primary key is thus (StudentID, 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>StudentID</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123456789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123456789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987654321</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>StudentID</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123456789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123456789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987654321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>StudentID</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123456789</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987654321</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?)
RELATIONAL SCHEMA DESIGN
(OR LOGICAL DESIGN)

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

\[ A_1, A_2, ..., A_n \]

then they must also agree on the attributes

\[ B_1, B_2, ..., B_m \]

Formally:

\[ A_1, A_2, ..., A_n \rightarrow B_1, B_2, ..., B_m \]
**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></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.
**FUNCTIONAL DEPENDENCIES (FDS)**

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

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

Logically equivalent:

\[
\neg \exists t, t' \in R, \quad (t.A_1 = t'.A_1 \land \ldots \land t.A_m = t'.A_m) \land \neg (t.B_1 = t'.B_1 \land \ldots \land t.B_n = t'.B_n)
\]
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
<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>

**EXAMPLE**

But not Phone ↗ 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>
<tr>
<td>name</td>
<td>category</td>
<td>color</td>
<td>department</td>
<td>price</td>
</tr>
<tr>
<td>---------</td>
<td>-------------</td>
<td>--------</td>
<td>---------------------</td>
<td>-------</td>
</tr>
<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?
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>StudentID</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123456789</td>
<td>206-555-1234</td>
<td>Seattle</td>
</tr>
<tr>
<td>Fred</td>
<td>123456789</td>
<td>206-555-6543</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987654321</td>
<td>908-555-2121</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

FD: StudentID -> Name, City
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.
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:

$\text{name}^+ = \{\text{name, color}\}$
$\{\text{name, category}\}^+ = \{\text{name, category, color, department, price}\}$
$\text{color}^+ = \{\text{color}\}$
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:**

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

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

Hence: name, category \(\rightarrow\) color, department, price
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 \( \{A, B\}^+ \) 
\[ X = \{A, B, \} \]

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 \( \{A, F\}^+ \) \( X = \{A, F, \} \)
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 \( \{A,B\}^+ \) \hspace{1cm} X = \{A, B, C, D, E\}

Compute \( \{A, F\}^+ \) \hspace{1cm} X = \{A, F, B, C, D, E\}
EXAMPLE

In class:

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

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 \(X\):

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

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

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

- I.e., for which closure = all attributes

A key is a minimal superkey

- A superkey and for which no subset is a superkey
- (A key already determines everything, so any superset of key does too.)
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?
What is the key?

(name, category) + = { name, category, price, 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
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 → A is OK if X is a (super)key

X → A is not OK otherwise

• Need to decompose the table, but how?

Boyce-Codd Normal Form
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:

\[
\forall X, \text{ either } X^+ = X \text{ or } X^+ = [\text{all attributes}]
\]
BCNF DECOMPOSITION ALGORITHM

Normalize(R)

find X 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);
### EXAMPLE

The only key is: \{StudentID, PhoneNumber\}  
Hence StudentID \rightarrow Name, City is a “bad” dependency

In other words:  
StudentID+ = StudentID, Name, City and is neither StudentID nor All Attributes
EXAMPLE BCNF DECOMPOSITION

<table>
<thead>
<tr>
<th>Name</th>
<th>StudentID</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>123456789</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>987654321</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

Let’s check anomalies:
- Redundancy ?
- Update ?
- Delete ?
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
EXAMPLE BCNF DECOMPOSITION

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

SSN \rightarrow \text{name, age}

age \rightarrow \text{hairColor}

Iteration 1: \textbf{Person}: \text{SSN}^+ = \text{SSN, name, age, hairColor}

Decompose into: \textbf{P}(SSN, name, age, hairColor)

\textbf{Phone}(SSN, phoneNumber)
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

**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?
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

**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

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

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

**EXAMPLE: BCNF**

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

R(A,B,C,D)

A \rightarrow B
B \rightarrow C

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

R(A,B,C,D)

$A^+ = ABC \neq ABCD$

$R_1(A,B,C)$

$R_2(A,D)$

A $\rightarrow$ B
B $\rightarrow$ C
EXAMPLE: BCNF

R(A,B,C,D)

A → B
B → C

R(A,B,C,D)
A⁺ = ABC ≠ ABCD

R₁(A,B,C)
B⁺ = BC ≠ ABC

R₂(A,D)
R(A,B,C,D)

**EXAMPLE: BCNF**

A \rightarrow B
B \rightarrow C

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^+?