Database Management Systems
CS 457

Lecture 8: Design Theory
Logistics

• HW1
  • Progress?
  • Demo on 2/22
• Next lecture (2/22), Nevada Bound in Sacramento
  • TA
  • HW1 review
  • HW2 preview
  • Database architecture and storage
Database Design Process

Conceptual Model:

Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Physical storage details

Conceptual Schema

Physical Schema
Relational Schema Design

<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>

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 numbers?

<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 ?)
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

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

then they must also agree on the attributes

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

Formally:

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

\[ A_1\ldots A_n \text{ determines } B_1\ldots B_m \]
### Functional Dependencies (FDs)

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

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

<table>
<thead>
<tr>
<th>R</th>
<th>A_1</th>
<th>...</th>
<th>A_m</th>
<th>B_1</th>
<th>...</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>

**Diagram:**
- 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
Side Note

• Logical equivalence
  – \((A \rightarrow B)\) means \((\neg A \lor B)\)
    • Truth table
    • Discrete math class?
      – A, B, A->B, Not A or B
      – T, T, T, T
      – T, F, F, F
      – F, T, T, T
      – F, F, T, T
<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>

\[ \text{Position} \rightarrow \text{Phone} \]
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 → Position
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>99</td>
</tr>
</tbody>
</table>

Do all the FDs hold on this instance?

name $\rightarrow$ color
category $\rightarrow$ department
color, category $\rightarrow$ price
**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>Black</td>
<td>Toys</td>
<td>99</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?
Terminology

- 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 \( F \), we are **stating a constraint on** \( R \)
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, \( \{A_1, \ldots, A_n\}^+ \) = the set of attributes B
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

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

\textbf{Repeat until} \( X \) doesn’t change \textbf{do:}
\textbf{if} \( B_1, \ldots, B_n \rightarrow C \) \textbf{is a FD and} \( B_1, \ldots, B_n \) \textbf{are all in} \( X \)
\textbf{then} \( \text{add} \ C \text{ to} \ X \).

\textbf{Example:}

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

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

\textbf{Hence:} \hspace{1cm} \text{name, category} \rightarrow \text{color, department, price}
Example

In class:

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

\[ \text{Compute \{A,B\}+ X = \{A, B, } \]

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

\[ \text{A, B } \rightarrow \text{ C} \]
\[ \text{A, D } \rightarrow \text{ E} \]
\[ \text{B } \rightarrow \text{ D} \]
\[ \text{A, F } \rightarrow \text{ B} \]
Example

In class:

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

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

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

In class:

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

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

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

In class:

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

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 \(X\):

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

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

\[
\begin{align*}
AB & \rightarrow CD, \quad AD \rightarrow BC, \quad ABC \rightarrow D, \quad ABD \rightarrow C, \quad ACD \rightarrow B
\end{align*}
\]
Keys

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

• 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 only 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) + = { 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 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 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
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

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);
The only key is: \{SSN, PhoneNumber\}
Hence \textbf{SSN} \rightarrow \textbf{Name, City} is a “bad” dependency
In other words: \textbf{SSN}^+ = \textbf{SSN, Name, City} and is neither \textbf{SSN} nor \textbf{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 ?

CS 457 - Spring 2018
Find $X$ s.t.: $X \neq X^+$ and $X^+ \neq [\text{all attributes}]$

**Example BCNF Decomposition**

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

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

FSN \to \text{name, age}

age \to \text{hairColor}

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

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

\( \text{Phone(SSN, phoneNumber)} \)
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

R(A,B,C,D)

A → B
B → C

R(A,B,C,D)
Example: BCNF

Recall: find $X$ s.t.
$X \not\subseteq X^+ \not\subseteq [\text{all-attrs}]$
R(A,B,C,D)

Example: BCNF

R(A,B,C,D)

A⁺ = ABC ≠ ABCD
Example: BCNF

R(A,B,C,D)

A \rightarrow B
B \rightarrow C

A^+ = ABC \neq ABCD

R(A,B,C,D)

R_1(A,B,C)

R_2(A,D)
Example: BCNF

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

\[ A \rightarrow B \]
\[ B \rightarrow C \]

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

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

\[ R_2(A,D) \]

CS 457 - Spring 2018
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)
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

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

\[ S_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \quad \text{and} \quad S_2(A_1, \ldots, A_n, C_1, \ldots, C_p) \]

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

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