Introduction to Database Systems
CSE 444

Lectures 8 & 9
Database Design

October 12 & 15, 2007

Announcements/Reminders

• Homework 1: solutions are posted
• Homework 2: posted later today (due Sat., Oct. 20)
• Project Phase 1 due tomorrow, 9pm

Outline

• The relational data model: 3.1
• Functional dependencies: 3.4

Schema Refinements = Normal Forms

• 1st Normal Form = all tables are flat
• 2nd Normal Form = obsolete
• Boyce Codd Normal Form = will study
• 3rd Normal Form = see book

First Normal Form (1NF)

• A database schema is in First Normal Form if all tables are flat

Relational Schema Design

Conceptual Model:

Relational Model: plus FD’s

Normalization: eliminates anomalies
Data Anomalies

When a database is poorly designed we get anomalies:

**Redundancy**: data is repeated

**Update anomalies**: need to change in several places

**Delete anomalies**: may lose data when we don’t want

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

Relational Schema Design

Recall set attributes (persons with several phones):

<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

**Anomalies**:
- **Redundancy** = repeated data
- **Update anomalies** = Fred moves to “Bellevue”
- **Deletion anomalies** = Joe deletes his phone number: what is his city?

Relation Decomposition (or Logical Design)

Main idea:
- Start with some relational schema
- Find out its **functional dependencies**
- Use them to design a better relational schema

Functional Dependencies

- A form of constraint
  - hence, part of the schema
- Finding them is part of the database design
- Also used in normalizing the relations

**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:

\[ A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m \]
When Does an FD Hold

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

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

Examples

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 \( \rightarrow \) Name, Phone, Position

Position \( \rightarrow \) Phone

but not Phone \( \rightarrow \) Position

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>

Position \( \rightarrow \) Phone

but not Phone \( \rightarrow \) Position

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>

Name \( \rightarrow \) color, category \( \rightarrow \) department

color, category \( \rightarrow \) price

Does this instance satisfy all the FDs?

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>

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>Office-supp</td>
<td>59</td>
</tr>
</tbody>
</table>

What about this one?
An Interesting Observation

If all these FDs are true:

| name → color |
| category → department |
| color, category → price |

Then this FD also holds:

| name, category → price |

Why??

Goal: Find ALL Functional Dependencies

- Anomalies occur when certain “bad” FDs hold
- We know some of the FDs
- Need to find all FDs, then look for the bad ones

Armstrong’s Rules (1/3)

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

Is equivalent to

| A_1, A_2, ..., A_n → B_1 |
| A_1, A_2, ..., A_n → B_2 |
| . . . |
| A_1, A_2, ..., A_n → B_m |

Splitting rule and Combing rule

Armstrong’s Rules (1/3)

| A_i → A_j |

Trivial Rule

Where i = 1, 2, ..., n

Why?

Armstrong’s Rules (1/3)

Transitive Closure Rule

If

| A_1, A_2, ..., A_n → B_1, B_2, ..., B_m |

and

| B_1, B_2, ..., B_m → C_1, C_2, ..., C_p |

then

| A_1, A_2, ..., A_n → C_1, C_2, ..., C_p |

Why?
Example (continued)

Start from the following FDs:

1. name → color
2. category → department
3. color, category → price

Infer the following FDs:

Inferred FD | Which Rule did we apply?
---|---
4. name, category → name | 4. name, category → name  
5. name, category → color | Transitivity on 4, 1  
6. name, category → category | Trivial rule  
7. name, category → color, category | Split/combine on 5, 6  
8. name, category → price | Transitivity on 3, 7

THIS IS TOO HARD! Let’s see an easier way.

Example (continued)

Answers:

Inferred FD | Which Rule did we apply?
---|---
4. name, category → name | Trivial rule  
5. name, category → color | Transitivity on 4, 1  
6. name, category → category | Trivial rule  
7. name, category → color, category | Split/combine on 5, 6  
8. name, category → price | Transitivity on 3, 7

Example (continued)

Closure of a set of Attributes

**Given** a set of attributes $A_1, …, A_n$

The **closure**, $\{A_1, …, A_n\}^+ = \{B \mid A_1, …, A_n \rightarrow B\}$

Example:

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

Closures:

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

Example:

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

$A, B \rightarrow C$

$A, D \rightarrow E$

$B \rightarrow D$

$A, F \rightarrow B$

$X = \{A, B\}^+ = \{A, B\}$

$X = \{A, F\}^+ = \{A, F\}$

Example

In class:

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

$A, B \rightarrow C$

$A, D \rightarrow E$

$B \rightarrow D$

$A, F \rightarrow B$

Example

In class:

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

$A, B \rightarrow C$

$A, D \rightarrow E$

$B \rightarrow D$

$A, F \rightarrow B$

Example

Why Do We Need Closure

- With closure we can find all FD's easily

- To check if $X \rightarrow A$
  - Compute $X^+$
  - Check if $A \in X^+$

Example

In class:

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

$A, B \rightarrow C$

$A, D \rightarrow E$

$B \rightarrow D$

$A, F \rightarrow B$
Using Closure to Infer ALL FDs

Example:

\[
A, B \rightarrow C \\
A, D \rightarrow B \\
B \rightarrow D
\]

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

\[
A^+ = A, \quad B^+ = BD, \quad C^+ = C, \quad D^+ = D \\
AB^+ = ABCD, \quad AC^+ = AC, \quad AD^+ = ABCD, \\
\quad BC^+ = BCD, \quad BD^+ = BD, \quad CD^+ = CD
\]

\[
ABC^+ = ABD^+ = ACD^+ = ABCD \quad \text{(no need to compute – why ?)} \\
BCD^+ = BCD, \quad ABCD^+ = ABCD
\]

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

\[
AB \rightarrow CD, \quad AD \rightarrow BC, \quad ABC \rightarrow D, \quad ABD \rightarrow C, \quad ACD \rightarrow B
\]

Another Example

- Enrollment(student, major, course, room, time)
  - student \( \rightarrow \) major
  - major, course \( \rightarrow \) room
  - course \( \rightarrow \) time

What else can we infer? [in class, or at home]

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
  - i.e. set of attributes which is a superkey and for which no subset is a superkey

Computing (Super)Keys

- Compute \( X^+ \) for all sets \( X \)
- If \( X^+ = \) all attributes, then \( X \) is a key
- List only the minimal \( X \)'s

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 \( \rightarrow \) price
- category \( \rightarrow \) color

What is the key?

\((\text{name, category})^+ = \) name, category, price, color

Hence (name, category) is a key
Examples of Keys

**Enrollment(student, address, course, room, time)**

- student \(\rightarrow\) address
- room, time \(\rightarrow\) course
- student, course \(\rightarrow\) room, time

(find keys at home)

Eliminating Anomalies

**Main idea:**

- \(X \rightarrow A\) is OK if \(X\) is a (super)key
- \(X \rightarrow A\) is not OK otherwise

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

What the key?  
\{SSN, PhoneNumber\}  
Hence \(SSN \rightarrow\) Name, City is a “bad” dependency

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

\[AB \rightarrow C\]  
\[BC \rightarrow A\]  
\[A \rightarrow BC\]  
\[B \rightarrow AC\]

Boyce-Codd Normal Form

A simple condition for removing anomalies from relations:

A relation \(R\) is in BCNF if:

If \(A_1, ..., A_n \rightarrow B\) is a non-trivial dependency in \(R\), then \(\{A_1, ..., A_n\}\) is a superkey for \(R\)

In other words: there are no “bad” FDs

Equivalently:

\(\forall X, either (X^+ = X)\) or \((X^+ = all\ attributes)\)
BCNF Decomposition Algorithm

repeat
  choose \( A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n \) that violates BCNF
  split \( R \) into \( R_1(A_1, \ldots, A_m, B_1, \ldots, B_n) \) and \( R_2(A_1, \ldots, A_m, \{\text{others}\}) \)
  continue with both \( R_1 \) and \( R_2 \)
until no more violations

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>

In practice, we have a better algorithm (coming up)

Example Decomposition

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

Decompose in BCNF (in class):

SSN \( \rightarrow \) Name, City

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>

What are the keys?

<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.: \( X \neq X^+ \neq \text{[all attributes]} \)

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

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

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

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

\[ R_1(B, C) \]
\[ R_2(A, B) \]

What happens if in \( R \) we first pick \( B^+ ? \) Or \( AB^+ ? \)

Decompositions in General

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

\[ R_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \]
\[ R_2(A_1, \ldots, A_n, C_1, \ldots, C_p) \]

\( R_1 = \) projection of \( R \) on \( A_1, \ldots, A_n, B_1, \ldots, B_m \)
\( R_2 = \) projection of \( R \) on \( A_1, \ldots, A_n, C_1, \ldots, C_p \)

Theory of Decomposition

- Sometimes it is correct:

<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>

Lossless decomposition

- Sometimes it is not:

<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>

Incorrect Decomposition

- Sometimes it is not:

<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>

Lossy decomposition

Decompositions in General

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

\[ R_1(A_1, \ldots, A_n, B_1, \ldots, B_m) \]
\[ R_2(A_1, \ldots, A_n, C_1, \ldots, C_p) \]

If \( A_1, \ldots, A_n \rightarrow B_1, \ldots, B_m \)
Then the decomposition is lossless

Note: don’t need \( A_1, \ldots, A_n \rightarrow C_1, \ldots, C_p \)

BCNF decomposition is always lossless. WHY?