Lecture 09:
Friday, October 18, 2002

Outline

• Functional dependencies (3.4)
• Rules about FDs (3.5)
• Design of a Relational schema (3.6)

Functional Dependencies

Definition: A₁, ..., Aₘ → B₁, ..., Bₙ holds in R if:

∀t, t' ∈ R, (t.A₁=t'.A₁ ∧ ... ∧ t.Aₘ=t'.Aₘ ⇒ t.B₁=t'.B₁ ∧ ... ∧ t.Bₙ=t'.Bₙ)

Examples of Keys

• Product(name, price, category, color)
  name, category → price
  category → color

  Keys are: (name, category) and all supersets

• Enrollment(student, address, course, room, time)
  student → address
  room, time → course
  student, course → room, time

  Keys are: [in class]

Formal definition of a key

• A key is a set of attributes A₁, ..., Aₙ s.t. for any other attribute B, A₁, ..., Aₙ → B

• A minimal key is a set of attributes which is a key and for which no subset is a key

• Note: book calls them superkey and key

Finding the Keys of a Relation

Given a relation constructed from an E/R diagram, what is its key?

Rules:
1. If the relation comes from an entity set, the key of the relation is the set of attributes which is the key of the entity set.
Finding the Keys

Rules:
2. If the relation comes from a many-many relationship, the key of the relation is the set of all attribute keys in the relations corresponding to the entity sets.

Expressing Dependencies
Say: “the CreditCard determines the Person”

Inference Rules for FD’s

Inference Rules for FD’s (continued)

A₁, A₂, …, Aᵱ → Bᵱ, B₂, …, Bᵐ

Splitting rule
and Combing rule

A₁, A₂, …, Aᵱ → Bᵱ
A₁, A₂, …, Aᵲ → Bᵲ
...
A₁, A₂, …, Aᵱ → Bᵐ

Trivial Rule

A₁, A₂, …, Aᵱ → Aᵱ

where i = 1, 2, ..., n

Why?
Inference Rules for FD’s (continued)

Transitive Closure Rule

If \( A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m \)
and \( B_1, B_2, \ldots, B_m \rightarrow C_1, C_2, \ldots, C_p \)
then \( A_1, A_2, \ldots, A_n \rightarrow C_1, C_2, \ldots, C_p \)

Why?

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

What else can we infer? [in class]

Closure of a set of Attributes

Given a set of attributes \( \{A_1, \ldots, A_n\} \) and a set of dependencies \( S \).
Problem: find all attributes \( B \) such that:
any relation which satisfies \( S \) also satisfies:
\( A_1, \ldots, A_n \rightarrow B \)

The closure of \( \{A_1, \ldots, A_n\} \), denoted \( \{A_1, \ldots, A_n\}^+ \),
is the set of all such attributes \( B \)

Closure Algorithm

Start with \( X = \{A_1, \ldots, A_n\} \).
Repeat until \( X \) doesn’t change do:
if \( B_1, B_2, \ldots, B_n \rightarrow C \) is in \( S \), and
\( B_1, B_2, \ldots, B_n \) are all in \( X \), and
\( C \) is not in \( X \)
then
add \( C \) to \( X \).

Example

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

\begin{align*}
A & \rightarrow C \\
A & \rightarrow E \\
B & \rightarrow D \\
A & \rightarrow B
\end{align*}

Closure of \( \{A,B\} \): \( X = \{A, B, \} \)
Closure of \( \{A,F\} \): \( X = \{A, F, \} \)
Why Is the Algorithm Correct?

• Show the following by induction:
  - For every $B$ in $X$:
    - $A_1, \ldots, A_n \implies B$.
  - Initially $X = \{A_1, \ldots, A_n\}$ -- holds
  - Induction step: $B_1, \ldots, B_m$ in $X$
    - $A_1, \ldots, A_n \implies B_1, \ldots, B_m$
    - We also have $B_1, \ldots, B_m \implies C$
    - By transitivity we have $A_1, \ldots, A_n \implies C$
  - This shows that the algorithm is sound; need to show it is complete

Relational Schema Design
(or Logical Design)

Main idea:
• Start with some relational schema
• Find out its FD’s
• Use them to design a better relational schema

Relational Schema Design
(or Logical Design)

When a database is poorly designed we get anomalies:
• Redundancy: data is repeated
• Updated 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>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>

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>Joe</td>
<td>987-65-4321</td>
<td>908-555-1234</td>
<td>Westfield</td>
</tr>
</tbody>
</table>

Anomalies:
• Redundancy = repeat data
• Update anomalies = Fred moves to “Bellvue”
• Deletion anomalies = Fred drops all phone numbers: what is his city?

Relational Schema Design

Conceptual Model:

Relational Model:
plus FD’s

Normalization:
Eliminates anomalies
Decompositions in General

\[ R(A_1, ..., A_n) \]

Create two relations \( R_1(B_1, ..., B_m) \) and \( R_2(C_1, ..., C_p) \) such that: \( B_1, ..., B_m \cup C_1, ..., C_p = A_1, ..., A_n \)

and:

\[ R_1 = \text{projection of } R \text{ on } B_1, ..., B_m \]
\[ R_2 = \text{projection of } R \text{ on } C_1, ..., C_p \]

Incorrect Decomposition

• Sometimes it is incorrect:

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Camera</td>
<td>Gadget</td>
<td>29.99</td>
</tr>
<tr>
<td>Camera</td>
<td>OneClick</td>
<td>24.99</td>
</tr>
<tr>
<td>Gadgets</td>
<td>Gizmo</td>
<td>19.99</td>
</tr>
</tbody>
</table>

Cameras:
- Westfield: 908-555-2121, Fred
- Seattle: 206-555-1234, Joe

Gadgets:
- Westfield: 908-555-2121, Fred
- Seattle: 206-555-1234, Joe

When we put it back:
Cannot recover information.

Normal Forms

First Normal Form (1NF) = all attributes are atomic

Second Normal Form (2NF) = old and obsolete

Third Normal Form (3NF) = this lecture

Boyce-Codd Normal Form (BCNF) = this lecture

Others...

Boyce-Codd Normal Form

A simple condition for removing anomalies from relations:

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

Whenever there is a nontrivial dependency \( A_1, ..., A_n \rightarrow B \) in \( R \), \( \{A_1, ..., A_n\} \) is a key for \( R \)

In English (though a bit vague):

Whenever a set of attributes of \( R \) is determining another attribute, should determine \textbf{all} the attributes of \( R \).

Example

<table>
<thead>
<tr>
<th>Name</th>
<th>SSN</th>
<th>PhoneNumber</th>
<th>City</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fred</td>
<td>213-45-6789</td>
<td>206-555-3212</td>
<td>Seattle</td>
</tr>
<tr>
<td>Joe</td>
<td>213-45-6789</td>
<td>206-555-4567</td>
<td>Seattle</td>
</tr>
<tr>
<td>Jim</td>
<td>303-45-6789</td>
<td>503-555-2134</td>
<td>Woodfield</td>
</tr>
<tr>
<td>Joe</td>
<td>203-45-6789</td>
<td>503-555-1234</td>
<td>Woodfield</td>
</tr>
</tbody>
</table>

What are the dependencies?
SSN \( \rightarrow \) Name, City
What are the keys?
\{Name, SSN, PhoneNumber\}
Is it in BCNF?
Decompose it into BCNF

<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>

SSN $\rightarrow$ Name, City

Summary of BCNF Decomposition

Find a dependency that violates the BCNF condition:

$A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m$

Heuristics: choose $B_1, B_2, \ldots, B_m$ "as large as possible"

Decompose:

Is there a 2-attribute relation that is not in BCNF?

Continue until there are no BCNF violations left.

Example Decomposition

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

SSN $\rightarrow$ name, age

age $\rightarrow$ hairColor

Decompose in BCNF (in class):

Step 1: find all keys

Step 2: now decompose

Other Example

$R(A,B,C,D) \rightarrow A \rightarrow B, B \rightarrow C$

Key: A, D

Violations of BCNF: $A \rightarrow B, A \rightarrow C, A \rightarrow BC$

Pick $A \rightarrow BC$: split into $R_1(A,BC)$, $R_2(A,D)$

What happens if we pick $A \rightarrow B$ first?

Correct Decompositions

A decomposition is lossless if we can recover:

$R(A,B,C)$

$\rightarrow$ Decompose

$R_1(A,B)$, $R_2(A,C)$

$\rightarrow$ Recover

$R'(A,B,C)$ should be the same as $R(A,B,C)$

$R'$ is in general larger than $R$. Must ensure $R' = R$

Correct Decompositions

Given $R(A,B,C)$ s.t. $A \rightarrow B$, the decomposition into $R_1(A,B)$, $R_2(A,C)$ is lossless
3NF: A Problem with BCNF

<table>
<thead>
<tr>
<th>Unit</th>
<th>Company</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

FD's: Unit \(\rightarrow\) Company, Company, Product \(\rightarrow\) Unit
So, there is a BCNF violation, and we decompose.

<table>
<thead>
<tr>
<th>Unit</th>
<th>Company</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Unit</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>No FDs</td>
</tr>
</tbody>
</table>

So, there is a BCNF violation, and we decompose.

So What’s the Problem?

<table>
<thead>
<tr>
<th>Unit</th>
<th>Company</th>
<th>Unit</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td>Galaga99</td>
<td>UW</td>
<td>Galaga99</td>
<td>databases</td>
</tr>
<tr>
<td>Bingo</td>
<td>UW</td>
<td>Bingo</td>
<td>databases</td>
</tr>
</tbody>
</table>

No problem so far. All local FD's are satisfied.
Let's put all the data back into a single table again:

<table>
<thead>
<tr>
<th>Unit</th>
<th>Company</th>
<th>Product</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

| Galaga99 | UW | databases |
| Bingo    | UW | databases |

Violates the dependency: company, product \(\rightarrow\) unit!

Solution: 3rd Normal Form (3NF)

A simple condition for removing anomalies from relations:

A relation R is in 3rd normal form if:
Whenever there is a nontrivial dependency \(A_1, A_2, ..., A_n \rightarrow B\)
for R, then \(\{A_1, A_2, ..., A_n\}\) a super-key for R,
or B is part of a key.