Introduction to Management CSE 344

Lectures 16: Database Design

Relational Schema Design

Conceptual Model:

Relational Model: plus FD's

Normalization: Eliminates anomalies

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>

Anomalies:
- Redundancy = repeat data
- Update anomalies = what if Fred moves to “Bellevue”?
- Deletion anomalies = what if Joe deletes his phone number?

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>

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: \( A_1, A_2, ..., A_n \rightarrow B_1, B_2, ..., B_m \)

---

**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 \( \rightarrow \) Name, Phone, Position

Position \( \rightarrow \) Phone

but not Phone \( \rightarrow \) Position

---

**Example**

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>

But not Phone \( \rightarrow \) Position

---

**Example**

Functional Dependencies (FDs)

**Definition**

\( A_1, A_2, ..., A_n \rightarrow B_1, ..., B_m \) holds in \( R \) if:

\( \forall t, t' \in R, (t.A_1 = t'.A_1 \land ... \land t.A_n = t'.A_n \Rightarrow t.B_1 = t'.B_1 \land ... \land t.B_m = t'.B_m) \)

---

Magda Balazinska - CSE 344, Fall 2012
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-sup.</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 → color
category → department
color, category → price

Then this FD also holds:

name, category → price

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)

Is equivalent to

A₁, A₂, ..., Aᵣ → B₁, B₂, ..., Bᵥ

Splitting rule and Combing rule

Armstrong’s Rules (2/3)

A₁, A₂, ..., Aᵣ → Aᵢ

Trivial Rule

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

Why?
Transitive 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?

Example (continued)

Start from the following FDs:

<table>
<thead>
<tr>
<th>Inferred FD</th>
<th>Which Rule did we apply?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. name ( \rightarrow ) color</td>
<td>Trivial rule</td>
</tr>
<tr>
<td>2. category ( \rightarrow ) department</td>
<td>Transitivity on 1, 4</td>
</tr>
<tr>
<td>3. color, category ( \rightarrow ) price</td>
<td>Split/combine on 5, 6</td>
</tr>
</tbody>
</table>

Answers:

<table>
<thead>
<tr>
<th>Inferred FD</th>
<th>Which Rule did we apply?</th>
</tr>
</thead>
<tbody>
<tr>
<td>4. name, category ( \rightarrow ) name</td>
<td>Trivial rule</td>
</tr>
<tr>
<td>5. name, category ( \rightarrow ) color</td>
<td>Transitivity on 3, 7</td>
</tr>
<tr>
<td>6. name, category ( \rightarrow ) category</td>
<td>Trivial rule</td>
</tr>
<tr>
<td>7. name, category ( \rightarrow ) color, category</td>
<td>Split/combine on 5, 6</td>
</tr>
<tr>
<td>8. name, category ( \rightarrow ) price</td>
<td>Transitivity on 3, 7</td>
</tr>
</tbody>
</table>

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

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:

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

X=\(\{A_1, \ldots, A_n\}\).

Repeat until \( X \) doesn't change:

- if \( B_1, \ldots, B_i \rightarrow C \) is a FD and \( B_1, \ldots, B_i \) are all in \( X \)
- then add \( C \) to \( X \).

Example:

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

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

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

Practice at Home

Find all FD’s implied by:

\[ A, B \rightarrow C \]
\[ A, D \rightarrow B \]
\[ B \rightarrow D \]
Practice at Home

Find all FD’s implied by:
A, B → C
A, D → B
B → D

Step 1: Compute X⁺, for every X:
A⁺ = A, B⁺ = BD, C⁺ = C, D⁺ = D
AB⁺ = ABCD, AC⁺ = AC, AD⁺ = ABCD,
BC⁺ = BCD, BD⁺ = BD, CD⁺ = CD
AB⁺ = ABD⁺ = ACD⁺ = ABCD (no need to compute – why?)
B⁺ = BCD, ABD⁺ = ABCD

Step 2: Enumerate all FD’s X → Y, s.t. Y ⊆ X⁺ and X ∩ Y = Ø:
AB → CD, AD → BC, ABC → D, ABD → C, ACD → B

Keys

• A superkey is a set of attributes A₁, …, Aₙ s.t. for any other attribute B, we have A₁, …, Aₙ → 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 superkey
• List only the minimal X’s to get the keys

Example

Product(name, price, category, color)

name, category → price
category → color

What is the 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

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

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

or

\[
\begin{align*}
AB &\rightarrow C \\
BC &\rightarrow A \\
A &\rightarrow BC \\
B &\rightarrow AC
\end{align*}
\]

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

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 is the key?

(SSN, PhoneNumber) Hence SSN \(\rightarrow\) Name, City is a "bad" dependency

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

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

The only key is: (SSN, PhoneNumber)
Hence SSN \(\rightarrow\) Name, City is a "bad" dependency
In other words:
\( SSN^+ = \text{Name, City} \) and is neither SSN nor 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>

SSN → Name, City

Let's check anomalies:

- Redundancy?
- Update?
- Delete?

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)

What are the keys?

Find X s.t.: X #X* # [all attributes]

Example BCNF Decomposition

R(A,B,C,D)

Practice at Home

A → B
B → C

R(A,B,C,D)
A* = ABC ≠ ABCD

Note the keys!
Practice at Home

R(A,B,C,D)

A→B
B→C

What are the keys?
A
→
B
B
→
C

R(A,B,C,D)
A* = ABC ≠ ABCD

R₁(A,B,C)
B* = BC ≠ ABC

R₂(A,D)

R₃(B,C)
R₄(A,B)

What happens if in R we first pick B*? Or AB*?

Schema Refinements
= Normal Forms

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