Lecture 15: Functional Dependencies
First Normal Form (1NF)

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

---

**Student**

<table>
<thead>
<tr>
<th>Name</th>
<th>GPA</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>3.8</td>
<td>Math, DB, OS</td>
</tr>
<tr>
<td>Bob</td>
<td>3.7</td>
<td>DB, OS</td>
</tr>
<tr>
<td>Carol</td>
<td>3.9</td>
<td>Math, OS</td>
</tr>
</tbody>
</table>

**Takes**

<table>
<thead>
<tr>
<th>Student</th>
<th>Course</th>
</tr>
</thead>
<tbody>
<tr>
<td>Alice</td>
<td>Math</td>
</tr>
<tr>
<td>Carol</td>
<td>Math</td>
</tr>
<tr>
<td>Alice</td>
<td>DB</td>
</tr>
<tr>
<td>Bob</td>
<td>DB</td>
</tr>
<tr>
<td>Alice</td>
<td>OS</td>
</tr>
<tr>
<td>Carol</td>
<td>OS</td>
</tr>
</tbody>
</table>

**Course**

- Math
- DB
- OS

---

May need to add keys
Conceptual Schema Design

Conceptual Model:

Relational Model: plus FD’s
(FD = Functional Dependency)

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
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>413-555-1234</td>
<td>Amherst</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>413-555-6543</td>
<td>Amherst</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)

The above is in 1NF, but what is the problem with this schema?
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>413-555-1234</td>
<td>Amherst</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>413-555-6543</td>
<td>Amherst</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 “Boston”?
- Deletion anomalies = what if Joe deletes his phone number?
  (what if Joe had only one phone #)
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>413-555-1234</td>
<td>Amherst</td>
</tr>
<tr>
<td>Fred</td>
<td>123-45-6789</td>
<td>413-555-6543</td>
<td>Amherst</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 “Boston” (how ?)
- Easy to delete all Joe’s phone numbers (how ?)
Relational Schema Design (Logical Design)

- **Main idea:**
  - Start with some relational schema
  - Find out its functional dependencies (discussed next!)
  - 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
- Use them to normalize the relations
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:**

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

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>A₁</th>
<th>...</th>
<th>Aₘ</th>
<th>B₁</th>
<th>...</th>
<th>Bₙ</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>

\[ \text{if } t, t' \text{ agree here} \quad \text{then } t, t' \text{ agree here} \]
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 $\rightarrow$ 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>

\[
\text{Position} \rightarrow \text{Phone}
\]
### 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 $\rightarrow$ Position
Example

FD’s are constraints:
• On some instances they hold
• On others they don’t

name ➔ color
category ➔ department
color, category ➔ price

<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>

Does this instance satisfy all the FDs?
Example

FD’s are constraints:
• On some instances they hold
• On others they don’t

name → color
category → department
color, category → price

<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>Gizmo</td>
<td>Stationary</td>
<td>Blue</td>
<td>Supplies</td>
<td>59</td>
</tr>
</tbody>
</table>

What about this one?
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

Why ??
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)

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

Is equivalent to

\[ A_1, A_2, \ldots, A_n \rightarrow B_1 \]
\[ A_1, A_2, \ldots, A_n \rightarrow B_2 \]
\[ \ldots \ldots \]
\[ A_1, A_2, \ldots, A_n \rightarrow B_m \]
Armstrong’s Rules (2/3)

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

where \( i = 1, 2, \ldots, n \)

**Trivial Rule**

Why?

<table>
<thead>
<tr>
<th></th>
<th>( A_1 )</th>
<th>( \ldots )</th>
<th>( A_m )</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Armstrong’s Rules (3/3)

Transitive Rule

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

Why?
Illustration for Transitivity

<table>
<thead>
<tr>
<th></th>
<th>A₁</th>
<th>...</th>
<th>Aₘ</th>
<th>B₁</th>
<th>...</th>
<th>Bₘ</th>
<th>C₁</th>
<th>...</th>
<th>Cₚ</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>3</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Example (continued)

Start from the following FDs:

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

Infer the following FDs:

<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></td>
</tr>
<tr>
<td>5. name, category $\rightarrow$ color</td>
<td></td>
</tr>
<tr>
<td>6. name, category $\rightarrow$ category</td>
<td></td>
</tr>
<tr>
<td>7. name, category $\rightarrow$ color, category</td>
<td></td>
</tr>
<tr>
<td>8. name, category $\rightarrow$ price</td>
<td></td>
</tr>
</tbody>
</table>
Example (continued)

Answers:

<table>
<thead>
<tr>
<th>Inferred FD</th>
<th>Which Rule did we apply?</th>
</tr>
</thead>
<tbody>
<tr>
<td>1. name → color</td>
<td></td>
</tr>
<tr>
<td>2. category → department</td>
<td></td>
</tr>
<tr>
<td>3. color, category → price</td>
<td></td>
</tr>
<tr>
<td>4. name, category → name</td>
<td>Trivial</td>
</tr>
<tr>
<td>5. name, category → color</td>
<td>Transitivity on 4, 1</td>
</tr>
<tr>
<td>6. name, category → category</td>
<td>Trivial</td>
</tr>
<tr>
<td>7. name, category → color, category</td>
<td>Split/combine on 5, 6</td>
</tr>
<tr>
<td>8. name, category → 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 \( A_1, ..., A_n \)

The **closure**, \( \{A_1, ..., A_n\}^+ \) = the set of attributes \( B \) such that \( A_1, ..., A_n \rightarrow B \)

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

Closures:
- \( name^+ = \{name, color\} \)
- \( \{name, category\}^+ = \{name, category, color, department, price\} \)
- \( color^+ = \{color\} \)
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 \( X \).

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

Hence: \( \text{name, category} \rightarrow \text{color, department, price} \)

Example:

- name → color
- category → department
- color, category → 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\}^+ \quad X = \{A, B, \} \}

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

In class:

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

\[
\begin{array}{c}
A, B \rightarrow C \\
A, D \rightarrow E \\
B \rightarrow D \\
A, F \rightarrow B \\
\end{array}
\]

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)

\[
\begin{array}{l}
A, B \rightarrow C \\
A, D \rightarrow E \\
B \rightarrow D \\
A, F \rightarrow B \\
\end{array}
\]

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

Compute \(\{A, F\}^+\) \(\ X = \{A, F, B, C, D, E\}\)
Why Do We Need Closure

- With closure we can find all FD’s easily

- To check if $X \rightarrow A$
  - Compute $X^+$
  - Check if $A \in X^+$
Using Closure to Infer ALL FDs

Example:

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

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

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

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

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

Enrollment(student, major, course, room, time)

student $\rightarrow$ major
major, course $\rightarrow$ room
course $\rightarrow$ time

What else can we infer?