Functional dependencies

Motivation: ‘good’ tables

takes1 (ssn, c-id, grade, name, address)

‘good’ or ‘bad’?

<table>
<thead>
<tr>
<th>ssn</th>
<th>c-id</th>
<th>Grade</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>413</td>
<td>A</td>
<td>smith</td>
<td>Main</td>
</tr>
<tr>
<td>143</td>
<td>417</td>
<td>B</td>
<td>smith</td>
<td>Main</td>
</tr>
<tr>
<td>123</td>
<td>211</td>
<td>A</td>
<td>smith</td>
<td>Main</td>
</tr>
</tbody>
</table>
Functional dependencies

‘Bad’ – Q: why?

<table>
<thead>
<tr>
<th>c-id</th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>A</td>
<td>smith Main</td>
</tr>
<tr>
<td>123</td>
<td>B</td>
<td>smith Main</td>
</tr>
<tr>
<td>123</td>
<td>A</td>
<td>smith Main</td>
</tr>
</tbody>
</table>

A: Redundancy
– space
– inconsistencies
– insertion/deletion anomalies (later…)

Q: What caused the problem?

Functional dependencies

• A: ‘name’ depends on the ‘ssn’
• define ‘depends’

<table>
<thead>
<tr>
<th>c-id</th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>A</td>
<td>smith Main</td>
</tr>
<tr>
<td>123</td>
<td>B</td>
<td>smith Main</td>
</tr>
<tr>
<td>123</td>
<td>A</td>
<td>smith Main</td>
</tr>
</tbody>
</table>
Overview

- Functional dependencies
  - why
  - definition
  - Armstrong’s “axioms”
  - closure and cover

Functional dependencies

Definition: \( a \rightarrow b \)

‘a’ functionally determines ‘b’

<table>
<thead>
<tr>
<th>San</th>
<th>c-id</th>
<th>Grade</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>413</td>
<td>A</td>
<td>smith</td>
<td>Main</td>
</tr>
<tr>
<td>123</td>
<td>415</td>
<td>B</td>
<td>smith</td>
<td>Main</td>
</tr>
<tr>
<td>123</td>
<td>211</td>
<td>A</td>
<td>smith</td>
<td>Main</td>
</tr>
</tbody>
</table>

Informally: ‘if you know ‘a’, there is only one ‘b’ to match’
Functional dependencies

formally:

\[ X \rightarrow Y \Rightarrow (t_1[x] = t_2[x] \Rightarrow t_1[y] = t_2[y]) \]

if two tuples agree on the ‘X’ attribute, the *must* agree on the ‘Y’ attribute, too (eg., if ssn is the same, so should address)

Functional dependencies

• ‘X’, ‘Y’ can be sets of attributes
• Q: other examples??

<table>
<thead>
<tr>
<th>ssn</th>
<th>c-id</th>
<th>Grade</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>413</td>
<td>A</td>
<td>smith</td>
<td>Main</td>
</tr>
<tr>
<td>123</td>
<td>415</td>
<td>B</td>
<td>smith</td>
<td>Main</td>
</tr>
<tr>
<td>123</td>
<td>411</td>
<td>A</td>
<td>smith</td>
<td>Main</td>
</tr>
</tbody>
</table>

Functional dependencies

• ssn -> name, address
• ssn, c-id -> grade

<table>
<thead>
<tr>
<th>ssn</th>
<th>c-id</th>
<th>Grade</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>413</td>
<td>A</td>
<td>smith</td>
<td>Main</td>
</tr>
<tr>
<td>123</td>
<td>415</td>
<td>B</td>
<td>smith</td>
<td>Main</td>
</tr>
<tr>
<td>123</td>
<td>411</td>
<td>A</td>
<td>smith</td>
<td>Main</td>
</tr>
</tbody>
</table>
Overview

- Functional dependencies
  - why
  - definition
  - Armstrong’s “axioms”
  - closure and cover

Functional dependencies

Closure of a set of FD: all implied FDs - eg.:

- ssn -> name, address
- ssn, c-id -> grade

imply

- ssn, c-id -> grade, name, address
- ssn, c-id -> ssn

FDs - Armstrong’s axioms

Closure of a set of FD: all implied FDs - eg.:

- ssn -> name, address
- ssn, c-id -> grade

how to find all the implied ones, systematically?
FDs - Armstrong’s axioms

“Armstrong’s axioms” guarantee soundness and completeness:

• Reflexivity: \( Y \subseteq X \Rightarrow X \rightarrow Y \)
  eg., ssn, name \( \rightarrow \) ssn

• Augmentation \( X \rightarrow Y \Rightarrow XW \rightarrow YW \)
  eg., ssn->name then ssn.grade-> name.grade

• Transitivity

  \[
  \begin{align*}
  X & \rightarrow Y \\
  Y & \rightarrow Z \\
  \Rightarrow & \text{ } X \rightarrow Z
  \end{align*}
  \]

  ssn \( \rightarrow \) address
  address \( \rightarrow \) county-tax-rate
  THEN:
  ssn \( \rightarrow \) county-tax-rate

‘sound’ and ‘complete’
**FDs - Armstrong’s axioms**

Additional rules:

- **Union**
  
  \[
  \{X \rightarrow Y, X \rightarrow Z\} \Rightarrow X \rightarrow YZ
  \]

- **Decomposition**
  
  \[
  X \rightarrow YZ \Rightarrow \begin{cases} 
  X \rightarrow Y \\
  X \rightarrow Z \end{cases}
  \]

- **Pseudo-transitivity**
  
  \[
  \begin{align*}
  X & \rightarrow Y \\
  YW & \rightarrow Z
  \end{align*}
  \Rightarrow XW \rightarrow Z
  \]

---

**FDs - Armstrong’s axioms**

Prove ‘Union’ from three axioms:

\[
\begin{align*}
X \rightarrow Y, X \rightarrow Z & \Rightarrow X \rightarrow YZ \\
\end{align*}
\]
FDs - Armstrong’s axioms

Prove Pseudo-transitivity:
\[ Y \subseteq X \Rightarrow X \rightarrow Y \]
\[ X \rightarrow Y \Rightarrow XW \rightarrowYW \]
\[ X \rightarrow Y \quad Y \rightarrow Z \Rightarrow X \rightarrow Z \]
\[ X \rightarrow Y \quad YW \rightarrow Z \Rightarrow XW \rightarrow Z \]

FDs - Armstrong’s axioms

Prove Decomposition
\[ Y \subseteq X \Rightarrow X \rightarrow Y \]
\[ X \rightarrow Y \Rightarrow XW \rightarrow YW \]
\[ X \rightarrow Y \quad Y \rightarrow Z \Rightarrow X \rightarrow Z \]
\[ X \rightarrow YZ \Rightarrow X \rightarrow Y \quad X \rightarrow Z \]

Overview

- Functional dependencies
  - why
  - definition
  - Armstrong’s “axioms”
  - closure and cover
FDs - Closure F+

Given a set F of FD (on a schema)
F+ is the set of all implied FD. Eg.,
takes(ssn, c-id, grade, name, address)
  ssn, c-id -> grade
  ssn-> name, address

FDs - Closure F+

ssn, c-id -> grade
ssn-> name, address
ssn-> ssn
ssn, c-id-> address
c-id, address-> c-id
...

FDs - Closure A+

Given a set F of FD (on a schema)
A+ is the set of all attributes determined by A:
takes(ssn, c-id, grade, name, address)
  ssn, c-id -> grade
  ssn-> name, address

{ssn}+ =??
FDs - Closure A+

* takes(ssn, c-id, grade, name, address) *
  
  * ssn, c-id -> grade *
  
  * ssn -> name, address  

\{ssn\}+ = \{ssn, name, address \}

FDs - Closure A+

* takes(ssn, c-id, grade, name, address) *
  
  * ssn, c-id -> grade *
  
  * ssn -> name, address  

\{c-id\}+ = ??

FDs - Closure A+

* takes(ssn, c-id, grade, name, address) *
  
  * ssn, c-id -> grade *
  
  * ssn -> name, address  

\{c-id, ssn\}+ = ??
FDs - Closure $A^+$

if $A^+ = \{\text{all attributes of table}\}$
then ‘$A$’ is a superkey

FDs - $A^+$ closure - not in book

Diagrams

AB$\rightarrow$C (1)  
A$\rightarrow$BC (2)  
B$\rightarrow$C (3)  
A$\rightarrow$B (4)

FDs - ‘canonical cover’ $F_c$

Given a set $F$ of FD (on a schema)  
$F_c$ is a minimal set of equivalent FD. Eg.,
takes(ssn, c-id, grade, name, address)  
ssn, c-id$\rightarrow$ grade  
ssn$\rightarrow$ name, address  
ssn,name$\rightarrow$ name, address  
ssn, c-id$\rightarrow$ grade, name
FDs - ‘canonical cover’ Fc

- why do we need it?
- define it properly
- compute it efficiently

- easier to compute candidate keys
FDs - ‘canonical cover’ Fc

• define it properly - three properties
  – 1) the RHS of every FD is a single attribute
  – 2) the closure of Fc is identical to the closure of F (ie., Fc and F are equivalent)
  – 3) Fc is minimal (ie., if we eliminate any attribute from the LHS or RHS of a FD, property #2 is violated

#3: we need to eliminate ‘extraneous’ attributes. An attribute is ‘extraneous if
  – the closure is the same, before and after its elimination
  – or if F-before implies F-after and vice-versa
FDs - ‘canonical cover’ $F_c$

Algorithm:
• examine each FD; drop extraneous LHS or RHS attributes; or redundant FDs
• make sure that FDs have a single attribute in their RHS
• repeat until no change

Trace algo for
$AB \rightarrow C$ (1)
$A \rightarrow BC$ (2)
$B \rightarrow C$ (3)
$A \rightarrow B$ (4)

split (2):
$AB \rightarrow C$ (1)
$A \rightarrow B$ (2')
$A \rightarrow C$ (2'')
$B \rightarrow C$ (3)
$A \rightarrow B$ (4)
### FDs - ‘canonical cover’ $F_c$

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$AB \rightarrow C$ (1)</td>
<td>$AB \rightarrow C$ (1)</td>
<td>$A \rightarrow B$ (2)</td>
<td>$A \rightarrow C$ (2'')</td>
<td>$B \rightarrow C$ (3)</td>
</tr>
</tbody>
</table>

$(2'')$: redundant (implied by (4), (3) and transitivity)

### FDs - ‘canonical cover’ $F_c$

<p>| | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$AB \rightarrow C$ (1)</td>
<td>$AB \rightarrow C$ (1)</td>
<td>$A \rightarrow C$ (2'')</td>
<td>$B \rightarrow C$ (3)</td>
<td>$A \rightarrow B$ (4)</td>
</tr>
</tbody>
</table>

$(2'')$: redundant (implied by (4), (3) and transitivity)

### FDs - ‘canonical cover’ $F_c$

<p>| | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>$AB \rightarrow C$ (1)</td>
<td>$B \rightarrow C$ (1')</td>
<td>$B \rightarrow C$ (3)</td>
<td>$A \rightarrow B$ (4)</td>
</tr>
</tbody>
</table>

in (1), 'A' is extraneous: (1),(3),(4) imply (1'),(3),(4), and vice versa
FDs - ‘canonical cover’ Fc

- nothing is extraneous
- all RHS are single attributes
- final and original set of FDs are equivalent (same closure)

Overview - conclusions

- Functional dependencies
  - why
  - definition
  - Armstrong’s “axioms”
  - closure and cover