Functional dependencies

- motivation: ‘good’ tables

```sql
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>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>
Functional dependencies

‘Bad’ – Q: why?

<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>211</td>
<td>A</td>
<td>smith</td>
<td>Main</td>
</tr>
</tbody>
</table>

Functional Dependencies

• A: Redundancy
  – space
  – inconsistencies
  – insertion/deletion anomalies (later…)
• Q: What caused the problem?

Functional dependencies

• A: ‘name’ depends on the ‘ssn’
• define ‘depends’
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>id</th>
<th>c-id</th>
<th>Grade</th>
<th>Name</th>
<th>Address</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>23</td>
<td>A</td>
<td>smith</td>
<td>Main</td>
</tr>
<tr>
<td>1</td>
<td>23</td>
<td>B</td>
<td>smith</td>
<td>Main</td>
</tr>
<tr>
<td>1</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:

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

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)

- ‘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>211</td>
<td>A</td>
<td>smith</td>
<td>Main</td>
</tr>
</tbody>
</table>

Functional dependencies

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

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

FDs - Armstrong’s axioms

- Transitivity

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

ssn -> address
address -> county-tax-rate

THEN:

ssn -> county-tax-rate

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

Additional rules:

• Union
  \[ X \rightarrow Y \ \text{and} \ \ X \rightarrow Z \implies X \rightarrow YZ \]

• Decomposition
  \[ X \rightarrow YZ \implies X \rightarrow Y \ \text{and} \ \ X \rightarrow Z \]

• Pseudo-transitivity
  \[ X \rightarrow Y \quad \text{and} \quad YW \rightarrow Z \implies XW \rightarrow Z \]

FDs - Armstrong’s axioms

Prove ‘Union’ from three axioms:

\[ X \rightarrow Y \quad \text{and} \quad X \rightarrow Z \implies X \rightarrow YZ \]
FDs - Armstrong’s axioms

Prove Pseudo-transitivity:

\[ Y \subseteq X \Rightarrow X \rightarrow Y \]
\[ X \rightarrow Y \Rightarrow XW \rightarrowYW \]
\[ X \rightarrow Y \]
\[ Y \rightarrow Z \] \Rightarrow X \rightarrow Z

\[ X \rightarrow Y \]
\[ YW \rightarrow Z \] \Rightarrow 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

\[ F \]

**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

\[ \{ \text{ssn}\} + =?? \]
FDs - Closure A+

takes(ssn, c-id, grade, name, address)
ssn, c-id -> grade  \{F\}
ssn-> name, address  \{F\}

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

FDs - Closure A+

takes(ssn, c-id, grade, name, address)
ssn, c-id -> grade  \{F\}
ssn-> name, address  \{F\}

\{{\text{c-id}}\}^+ = ??

FDs - Closure A+

takes(ssn, c-id, grade, name, address)
ssn, c-id -> grade  \{F\}
ssn-> name, address  \{F\}

\{{\text{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**

$A \rightarrow B\rightarrow C$ (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)

\[
\begin{align*}
\text{ssn, c-id} & \rightarrow \text{grade} \\
\text{ssn} & \rightarrow \text{name, address} \\
\text{ssn, name} & \rightarrow \text{name, address} \\
\text{ssn, c-id} & \rightarrow \text{grade, name}
\end{align*}
\]
FDs - ‘canonical cover’ \( F_c \)

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

\[ F_c : \]

- \( \text{ssn, c-id} \rightarrow \text{grade} \)
- \( \text{ssn} \rightarrow \text{name, address} \)
- \( \text{ssn, name} \rightarrow \text{name, address} \)
- \( \text{ssn, c-id} \rightarrow \text{name, grade, address} \)
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 (i.e., Fc and F are equivalent)
  – 3) Fc is minimal (i.e., 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

### Examples

<table>
<thead>
<tr>
<th>FDs</th>
<th>Canonical Cover</th>
</tr>
</thead>
<tbody>
<tr>
<td>ssn, c-id -&gt; grade</td>
<td>Fc</td>
</tr>
<tr>
<td>ssn-&gt; name, address</td>
<td></td>
</tr>
<tr>
<td>ssn,name-&gt; name, address</td>
<td></td>
</tr>
<tr>
<td>ssn, c-id-&gt; grade, name</td>
<td></td>
</tr>
</tbody>
</table>

F

FDs - ‘canonical cover’ Fc

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

FDs - ‘canonical cover’ Fc

Trace algo for
AB->C  (1)
A->BC  (2)
B->C  (3)
A->B  (4)

split (2):

AB->C  (1)
A->B  (2')
A->C  (2'')
B->C  (3)
A->B  (4)
FDs - ‘canonical cover’ $F_c$

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

- (2’): redundant (implied by (4), (3) and transitivity)

- (2’): redundant (implied by (4), (3) and transitivity)

- (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