Functional dependencies

- motivation: ‘good’ tables

takes1 \((\text{ssn}, \text{c-id}, \text{grade}, \text{name}, \text{address})\)

‘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?
• A: Redundancy
  – space
  – inconsistencies
  – insertion/deletion anomalies

Pitfalls

• insertion anomaly:
  – “jones” registers, but takes no class - no place to store his address!

Functional dependencies

‘Bad’ – Q: why?
• A: Redundancy
  – space
  – inconsistencies
  – insertion/deletion anomalies (later…)

Pitfalls

• deletion anomaly:
  – delete the last record of ‘smith’ (we lose his address!)

Q: What caused the problem?
Functional dependencies

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

<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>

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>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

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

<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

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 \( address \))

Functional dependencies

• ‘X’, ‘Y’ can be sets of attributes
• Q: other examples?? (no repeat courses)

Overview

• Functional dependencies
  – why
  – definition
  – Armstrong’s “axioms”
  – closure and cover
Overall goal for both lectures

- Given tables
  - STUDENT(ssn, ….)
  - TAKES( ssn, cid, …)
- And FD (ssn -> …, cid-> …)
- WRITE CODE
- To automatically generate ‘good’ schemas

“we want tables where the attributes depend on the primary key, on the whole key, and nothing but the key”

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
  \[
  \begin{align*}
  X \rightarrow Y \\
  Y \rightarrow Z
  \end{align*}
  \Rightarrow X \rightarrow Z
  \]
  ssn -> address
  address -> county-tax-rate
  THEN:
  ssn -> county-tax-rate

FDs - Armstrong’s axioms

- Additional rules:
  - Union
    \[
    \begin{align*}
    X \rightarrow Y \\
    X \rightarrow Z
    \end{align*}
    \Rightarrow X \rightarrow YZ
  \]
  - Decomposition
    \[
    X \rightarrow YZ \Rightarrow \begin{align*}
    X \rightarrow Y \\
    X \rightarrow Z
    \end{align*}
    \]
  - 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 ‘Union’ from three axioms:

\[
\begin{align*}
X \rightarrow Y \quad (1) \\
X \rightarrow Z \quad (2)
\end{align*}
\]

\( (1) + \text{augm. w/ } Z \Rightarrow XZ \rightarrow YZ \quad (3) \)

\( (2) + \text{augm. w/ } X \Rightarrow XX \rightarrow XZ \quad (4) \)

but \( XX \in X \); thus

\( (3) + (4) \text{ and transitivity } \Rightarrow X \rightarrow YZ \)

FDs - Armstrong’s axioms

Prove Pseudo-transitivity:

\[
\begin{align*}
Y \subseteq X & \Rightarrow X \rightarrow Y \\
X \rightarrow Y & \Rightarrow XW \rightarrowYW \\
X \rightarrow Y & \Rightarrow YW \rightarrow Z \\
Y \rightarrow Z \Rightarrow X \rightarrow Z
\end{align*}
\]

FDs - Armstrong’s axioms

Prove Decomposition

\[
\begin{align*}
Y \subseteq X & \Rightarrow X \rightarrow Y \\
X \rightarrow Y & \Rightarrow XW \rightarrowYW \\
X \rightarrow Y & \Rightarrow X \rightarrow Z \\
X \rightarrow Y & \Rightarrow X \rightarrow YZ
\end{align*}
\]
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 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+

\[ \text{takes}(\text{ssn}, \text{c-id}, \text{grade}, \text{name}, \text{address}) \]

= \{ \text{ssn, c-id} \rightarrow \text{grade} \}

= \{ \text{ssn} \rightarrow \text{name, address} \}

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

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

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

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

FDs - A+ closure - not in book

Diagrams

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

Paint ‘A’ ‘red’;
For each arrow, paint tip ‘red’, if base is ‘red’

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’ \( F_c \)

\[
\begin{align*}
\text{ssn, c-id} & \rightarrow \text{grade} \\
\text{ssn} & \rightarrow \text{name, address} \\
\text{ssn, name} & \rightarrow \text{name, address} \\
\text{ssn, c-id} & \rightarrow \text{grade, name}
\end{align*}
\]

\[
\text{\{ takes(ssn, c-id, grade, name, address) \}}
\]

- why do we need it?
  - easier to compute candidate keys
- define it properly
- compute it efficiently

FDs - ‘canonical cover’ \( F_c \)

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

FDs - ‘canonical cover’ \( F_c \)

- define it properly - three properties
  - 1) the RHS of every FD is a single attribute
  - 2) the closure of \( F_c \) is identical to the closure of \( F \) (ie., \( F_c \) and \( F \) are equivalent)
  - 3) \( F_c \) 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)
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’ Fc

AB->C (1)  
A->B (2')
A->C (2'')
B->C (3)
A->B (4)

split (2):

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

FDs - ‘canonical cover’ Fc

AB->C (1)  
A->B (2')
A->C (2'')
B->C (3)
A->B (4)

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

(1'),(3),(4), and vice versa
FDs - ‘canonical cover’ $F_c$

- $B \rightarrow C$ (1)
- $B \rightarrow C$ (3)
- $A \rightarrow B$ (4)

- 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