CMU SCS

Carnegie Mellon Univ. Dept. of Computer Science 15-415/615 - DB Applications

Lecture #16: Schema Refinement & Normalization - Functional Dependencies

(R&G, ch. 19)

CMU SCS

Functional dependencies

- motivation: 'good' tables

takes1 (<u>ssn, c-id</u>, grade, name, address)

'good' or 'bad'?

CMU SCS

Functional dependencies

takes1 (<u>ssn, c-id</u>, grade, name, address)

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

CMU SCS

Functional dependencies

'Bad' – Q: why?

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

CMU SCS

Functional dependencies

'Bad' – Q: why?

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

- A: Redundancy
  - space
  - inconsistencies
  - insertion/deletion anomalies

CMU SCS

Pitfalls

- insertion anomaly:
  - "jones" registers, but takes no class - no place to store his address!

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
...	...	...	...	...
234	null	null	jones	Forbes

CMU SCS

Pitfalls

- deletion anomaly:
  - delete the last record of 'smith' (we lose his address!)

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

CMU SCS

Functional dependencies

'Bad' – Q: why?

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

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

CMU SCS

Functional dependencies

- A: 'name' depends on the 'ssn'
- define 'depends'

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

CMU SCS

Overview

- Functional dependencies
  - why
  - definition
  - Armstrong's "axioms"
  - closure and cover

CMU SCS

Functional dependencies

Definition: $a \rightarrow b$

'a' functionally determines 'b'

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

CMU SCS

Functional dependencies

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

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

Functional dependencies

formally:

$$X \rightarrow Y \quad \Rightarrow \quad (t1[x] = t2[x] \Rightarrow t1[y] = t2[y])$$

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)

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

Functional dependencies

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

Ssn	c-id	Grade	Name	Address
123	413	A	smith	Main
123	415	B	smith	Main
123	211	A	smith	Main

Overview

- Functional dependencies
  - why
  - definition
  - Armstrong's "axioms"
  - closure and cover

CMU SCS

Overall goal for both lectures

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

Faloutsos & Pavlo SCS 15-415/615 17

CMU SCS

Overall goal for both lectures

"we want tables where the attributes depend on the primary key, on the **whole** key, and **nothing but** the key"

Faloutsos & Pavlo CMU SCS 15-415/615 18

CMU SCS

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

Faloutsos & Pavlo SCS 15-415/615 19

CMU SCS

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?

Faloutsos & Pavlo SCS 15-415/615 20

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 $\left.\begin{matrix} X \rightarrow Y \\ Y \rightarrow Z \end{matrix}\right\} \Rightarrow X \rightarrow Z$

  ssn -> address

  address -> county-tax-rate

  THEN:

  ssn -> county-tax-rate

FDs - Armstrong's axioms

Reflexivity: $Y \subseteq X \Rightarrow X \rightarrow Y$

Augmentation: $X \rightarrow Y \Rightarrow XW \rightarrow YW$

Transitivity: $\left.\begin{matrix} X \rightarrow Y \\ Y \rightarrow Z \end{matrix}\right\} \Rightarrow X \rightarrow Z$

'sound' and 'complete'

FDs - Armstrong's axioms

Additional rules:

- Union $\left.\begin{matrix} X \rightarrow Y \\ X \rightarrow Z \end{matrix}\right\} \Rightarrow X \rightarrow YZ$
- Decomposition $X \rightarrow YZ \Rightarrow \left.\begin{matrix} X \rightarrow Y \\ X \rightarrow Z \end{matrix}\right\}$
- Pseudo-transitivity

  $\left.\begin{matrix} X \rightarrow Y \\ YW \rightarrow Z \end{matrix}\right\} \Rightarrow XW \rightarrow Z$

CMU SCS

FDs - Armstrong's axioms

Prove 'Union' from three axioms:

$$\left.\begin{array}{l} X \rightarrow Y \\ X \rightarrow Z \end{array}\right\} \stackrel{?}{\Rightarrow} X \rightarrow YZ$$

Faloutsos & Pavlo SCS 15-415/615 25

CMU SCS

FDs - Armstrong's axioms

Prove 'Union' from three axioms:

$$\left.\begin{array}{ll} X \rightarrow Y & (1) \\ X \rightarrow Z & (2) \end{array}\right\}$$

$$(1) + augm.\ w/Z \Rightarrow XZ \rightarrow YZ\ (3)$$

$$(2) + augm.\ w/X \Rightarrow XX \rightarrow XZ (4)$$

$$but\ XX is\ X; thus$$

$$(3) + (4)\ and\ transitivity \Rightarrow X \rightarrow YZ$$

Faloutsos & Pavlo SCS 15-415/615 26

CMU SCS

FDs - Armstrong's axioms

Prove Pseudo-transitivity:

$$Y \subseteq X \Rightarrow X \rightarrow Y$$

$$X \rightarrow Y \Rightarrow XW \rightarrow YW$$

$$\left.\begin{array}{l} X \rightarrow Y \\ Y \rightarrow Z \end{array}\right\} \Rightarrow X \rightarrow Z$$

$$\left.\begin{array}{l} X \rightarrow Y \\ YW \rightarrow Z \end{array}\right\} \stackrel{?}{\Rightarrow} XW \rightarrow Z$$

Faloutsos & Pavlo SCS 15-415/615 27

CMU SCS

FDs - Armstrong's axioms

Prove Decomposition

$$Y \subseteq X \Rightarrow X \rightarrow Y$$

$$X \rightarrow Y \Rightarrow XW \rightarrow YW$$

$$\left.\begin{array}{l} X \rightarrow Y \\ Y \rightarrow Z \end{array}\right\} \Rightarrow X \rightarrow Z$$

$$X \rightarrow YZ \stackrel{?}{\Rightarrow} \left.\begin{array}{l} X \rightarrow Y \\ X \rightarrow Z \end{array}\right\}$$

Faloutsos & Pavlo SCS 15-415/615 28

CMU SCS

Overview

- Functional dependencies
  - why
  - definition
  - Armstrong's "axioms"
  - closure and cover

CMU SCS

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

CMU SCS

FDs - Closure F+

ssn, c-id -> grade

ssn-> name, address

ssn-> ssn

ssn, c-id-> address

c-id, address-> c-id

...

} F+

CMU SCS

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

} F

{ssn}+ =??

CMU SCS

FDs - Closure A+

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

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

{ssn}+ ={ssn,

Faloutsos & Pavlo SCS 15-415/615 33

CMU SCS

FDs - Closure A+

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

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

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

Faloutsos & Pavlo SCS 15-415/615 34

CMU SCS

FDs - Closure A+

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

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

{c-id}+ = ??

Faloutsos & Pavlo SCS 15-415/615 35

CMU SCS

FDs - Closure A+

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

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

{c-id, ssn}+ = ??

Faloutsos & Pavlo SCS 15-415/615 36

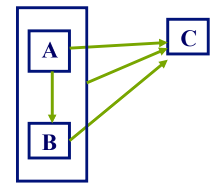
FDs - Closure A+

if A+ = {all attributes of table}
then ‘A’ is a **superkey**

FDs - A+ closure - not in book

Diagrams

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



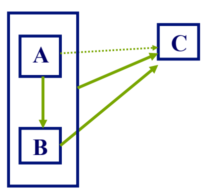
Paint ‘A’ ‘red’;

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

FDs - A+ closure - not in book

Diagrams

AB->C (1)
[redacted]
B->C (3)
A->B (4)



Repeat, without fd (2):

Paint ‘A’ ‘red’;

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

FDs - ‘canonical cover’ Fc

Given a set F of FD (on a schema)
Fc is a minimal set of equivalent FD. Eg.,
takes(ssn, c-id, grade, name, address)

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

} **F**

CMU SCS

FDs - 'canonical cover' Fc

Fc

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

F

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

Faloutsos & Pavlo SCS 15-415/615 41

CMU SCS

FDs - 'canonical cover' Fc

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

Faloutsos & Pavlo SCS 15-415/615 42

CMU SCS

FDs - 'canonical cover' Fc

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

Faloutsos & Pavlo SCS 15-415/615 43

CMU SCS

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

Faloutsos & Pavlo SCS 15-415/615 44

CMU SCS

FDs - ‘canonical cover’ Fc

#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

Faloutsos & Pavlo SCS 15-415/615 45

CMU SCS

FDs - ‘canonical cover’ Fc

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

} F

Faloutsos & Pavlo SCS 15-415/615 46

CMU SCS

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

Faloutsos & Pavlo SCS 15-415/615 47

CMU SCS

FDs - ‘canonical cover’ Fc

Trace algo for

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

Faloutsos & Pavlo SCS 15-415/615 48

CMU SCS

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)

CMU SCS

FDs - 'canonical cover' Fc

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

AB->C (1)

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

CMU SCS

FDs - 'canonical cover' Fc

AB->C (1)

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

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

AB->C (1)

B->C (3)
A->B (4)

CMU SCS

FDs - 'canonical cover' Fc

AB->C (1)

B->C (3)
A->B (4)

in (1), 'A' is extraneous: (1),(3),(4) imply (1'),(3),(4), and vice versa

B->C (1')

B->C (3)
A->B (4)

CMU SCS

FDs - 'canonical cover' Fc

~~B->C (1')~~

B->C (3)
A->B (4)

B->C (3)
A->B (4)

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

Faloutsos & Pavlo SCS 15-415/615 53

CMU SCS

FDs - 'canonical cover' Fc

BEFORE	AFTER
AB->C (1)	
A->BC (2)	
B->C (3)	B->C (3)
A->B (4)	A->B (4)

R(A,B,C)

Faloutsos & Pavlo SCS 15-415/615 54

CMU SCS

Overview - conclusions

- Functional dependencies
  - why
  - definition
  - Armstrong's "axioms"
  - closure and cover

Faloutsos & Pavlo SCS 15-415/615 55