CIS 560 – Database System Concepts

Lecture 10

Functional Dependencies and Normalization

September 18, 2013

Credits for slides: Suciu, Chang, Ullman.

Announcements

- HW3 due Friday
- HW4 will be posted Friday, due September 27th
- Exam 1 – October 7th (sample exam posted on KSOL)
- Project information posted on KSOL
- Project proposals
  - Information about team members and English description due September 24th
  - E/R diagram and relational schema due October 4th
- Proposal presentations October 9th and 11th

Outline

Last time:

- DB Design: Functional Dependencies (3.1 – 3.2)

Today:

- DB Design: Functional Dependencies (3.1 – 3.2)
- DB Design: Normalization (3.3-3.4)

Next:

- Transactions in SQL

3

Review

- Data anomalies?
- Functional dependencies?
- Armstrong's rules?

Goal: 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
- With *closure* we can find all FD’s easily

Closure of a set of Attributes

Given a set of attributes $A_1, \ldots, A_n$

The **closure**, $\{A_1, \ldots, A_n\}^+$ = the set of attributes B s.t. $A_1, \ldots, A_n \rightarrow B$

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.

Example:

name → color
category → department
color, category → price

$\{name, category\}^+ =$
{ }

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.

Example:

name → color
category → department
color, category → price

$\{name, category\}^+ =$
{ name, category, color, department, price }

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.

Example:

name → color
category → department
color, category → price

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

Hence: name, category → color, department, price

Example

In class:

R(A,B,C,D,E,F)

A, B → C
A, D → E
B → D
A, F → B

Compute $\{A,B\}^+$ X = {A, B, }

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 FDs easily
- To check if $X \rightarrow A$
  - Compute X^+
  - Check if $A \in X^+$

Using Closure to Infer ALL FDs

Example:

$A, B \rightarrow C$
$A, D \rightarrow B$
$B \rightarrow D$

Step 1: Compute X^+, for every X:

Step 2: Enumerate all FD's $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \varnothing$:

Using Closure to Infer ALL FDs

Example:

$A, B \rightarrow C$
$A, D \rightarrow B$
$B \rightarrow D$

Step 1: Compute X^+, for every X:

$A^+ = A$, $B^+ = BD$, $C^+ = C$, $D^+ = D$
$AB^+ = ABCD$, $AC^+ = AC$, $AD^+ = ABCD$,
$BC^+ = BCD$, $BD^+ = BD$, $CD^+ = CD$
$ABC^+ = ABD^+ = ACD^+ = ABCD$ (no need to compute– why ?)
$BCD^+ = BCD$, $ABCD^+ = ABCD$

Step 2: Enumerate all FD's $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \varnothing$:

Using Closure to Infer ALL FDs

Example:

A, B	$\rightarrow$	C
A, D	$\rightarrow$	B
B	$\rightarrow$	D

Step 1: Compute X^+, for every X:

$A^+ = A$, $B^+ = BD$, $C^+ = C$, $D^+ = D$
$AB^+ = ABCD$, $AC^+ = AC$, $AD^+ = ABCD$,
$BC^+ = BCD$, $BD^+ = BD$, $CD^+ = CD$
$ABC^+ = ABD^+ = ACD^+ = ABCD$ (no need to compute– why ?)
$BCD^+ = BCD$, $ABCD^+ = ABCD$

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

$AB \rightarrow CD$, $AD \rightarrow BC$, $BC \rightarrow D$, $ABC \rightarrow D$, $ABD \rightarrow C$, $ACD \rightarrow B$

Another Example

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

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

What else can we infer ? [at home]

Keys

- A **superkey** is a set of attributes $A_1, \ldots, A_n$ s.t. for any other attribute B, we have $A_1, \ldots, A_n \rightarrow B$
- A **key** is a minimal superkey
  - I.e. set of attributes which is a superkey and for which no subset is a superkey

Computing (Super)Keys

- Compute X^+ for all sets X
- If X^+ = all attributes, then X is a (super)key
- List only the minimal X's

Example

Product(name, price, category, color)

name, category $\rightarrow$ price
category $\rightarrow$ color

What is the key?

Example

Product(name, price, category, color)

name, category $\rightarrow$ price
category $\rightarrow$ color

What is the key ?

$(\text{name, category})^{+} = \{\text{name, category, price, color}\}$

Hence (name, category) is a key

Examples of Keys

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

student → address
room, time → course
student, course → room, time

Examples of Keys

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

student → address
room, time → course
student, course → room, time

Keys: {student, room, time}, {student, course} and all supersets

Key or Keys?

Can we have more than one key?

Given R(A,B,C) define FD's s.t. there are two or more keys

Key or Keys?

Can we have more than one key?

Given R(A,B,C) define FD's s.t. there are two or more keys

$AB \rightarrow C$ $BC \rightarrow A$	or	$A \rightarrow BC$ $B \rightarrow AC$

What are the keys here?

Can you design FDs such that there are *three* keys?

Eliminating Anomalies

Main idea:

- $X \to A$ is OK if X is a (super)key
- $X \to A$ is not OK otherwise

Example

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Topeka
Fred	123-45-6789	206-555-6543	Topeka
Joe	987-65-4321	908-555-2121	Manhattan
Joe	987-65-4321	908-555-1234	Manhattan

SSN → Name, City

What is the key?

{SSN, PhoneNumber}

Hence SSN → Name, City is a "bad" dependency

Boyce-Codd Normal Form

A simple condition for removing anomalies from relations:

A relation R is in BCNF if:

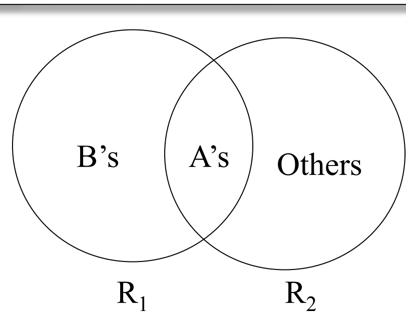
If $A_1, ..., A_n \rightarrow B$ is a non-trivial dependency in R, then $\{A_1, ..., A_n\}$ is a superkey for R

In other words: there are no "bad" FDs

Equivalently:
$\forall$ X, either $(X^+ = X)$ or $(X^+ = \text{all attributes})$

BCNF Decomposition Algorithm

repeat
choose $A_1, \ldots, A_m \rightarrow B_1, \ldots, B_n$ that violates BNCF
split R into $R_1(A_1, \ldots, A_m, B_1, \ldots, B_n)$ and $R_2(A_1, \ldots, A_m, [\text{others}])$
continue with both R_1 and R_2
until no more violations



Is there a 2-attribute relation that is not in BCNF?

In practice, we have a better algorithm (coming up)

Example

Name	SSN	PhoneNumber	City
Fred	123-45-6789	206-555-1234	Topeka
Fred	123-45-6789	206-555-6543	Topeka
Joe	987-65-4321	908-555-2121	Manhattan
Joe	987-65-4321	908-555-1234	Manhattan

SSN → Name, City

What is the key?

{SSN, PhoneNumber}

use SSN → Name, City to split

Example

Name	SSN	City
Fred	123-45-6789	Topeka
Joe	987-65-4321	Manhattan

SSN → Name, City

SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121
987-65-4321	908-555-1234

Let's check anomalies:

- Redundancy?
- Update?
- Delete?

BCNF Decomposition Algorithm

BCNF_Decompose(R)

find X s.t.: $X \neq X^+ \neq$ [all attributes]

<u>if</u> (not found) **<u>then</u>** "R is in BCNF"

<u>let</u> $Y = X^+ - X$
<u>let</u> $Z =$ [all attributes] $- X^+$
decompose R into $R1(X \cup Y)$ and $R2(X \cup Z)$
continue to decompose recursively R1 and R2

Find X s.t.: $X \neq X^+ \neq$ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age
age → hairColor

Find X s.t.: $X \neq X^+ \neq$ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age

age → hairColor

Iteration 1: Person

SSN^+ = {SSN, name, age, hairColor}

Decompose into: P(SSN, name, age, hairColor)

Phone(SSN, phoneNumber)

Find X s.t.: $X \neq X^+ \neq$ [all attributes]

Example BCNF Decomposition

Person(name, SSN, age, hairColor, phoneNumber)

SSN → name, age

age → hairColor

Iteration 1: Person

SSN^+ = {SSN, name, age, hairColor}

Decompose into: P(SSN, name, age, hairColor)

Phone(SSN, phoneNumber)

Iteration 2: P

age^+ = {age, hairColor}

Decompose: People(SSN, name, age)

Hair(age, hairColor)

Phone(SSN, phoneNumber)