Lecture 9: Database Design

Wednesday, January 25, 2006

1

Closure of a set of Attributes

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

The **closure**, $\{A_1, \ldots, A_n\}^+$, is 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}

2

Closure Algorithm

Start with X={A1, …, An}.

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 }

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, }

Why Do We Need Closure

- With closure we can find all FD's easily
- To check if $X \rightarrow A$
  - Compute X^+
  - Check if $A \in X^+$

5

Using Closure to Infer ALL FDs

Example:

A, B	→	C
A, D	→	B
B	→	D

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

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

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

AB → CD, AD→BC, ABC → D, ABD → C, ACD → B

6

Another Example

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

  student → major

  major, course → room

  course → time

  What else can we infer ? [in class, or at home]

7

Back to Conceptual Design

Now we know how to find more FDs, it's easy

- Search for "bad" FDs
- If there are such, then decompose the table into two tables, repeat for the subtables.
- When done, the database schema is *normalized*

Unfortunately, there are several normal forms…

8

Normal Forms

First Normal Form = all attributes are atomic

Second Normal Form (2NF) = old and obsolete

Third Normal Form (3NF) = will discuss

Boyce Codd Normal Form (BCNF) = will discuss

Others...

Keys

- A **superkey** is a set of attributes $A_1, ..., A_n$ s.t. for any other attribute B, we have $A_1, ..., A_n \rightarrow B$
- 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 key
- List only the minimal X's

11

Example

Product(name, price, category, color)

name, category → price
category → color

What is the key ?

12

Example

Product(name, price, category, color)

name, category → price
category → color

What is the key ?

(name, category) + = name, category, price, color

Hence (name, category) is a key

13

Examples of Keys

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

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

(find keys at home)

14

Eliminating Anomalies

Main idea:

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

15

Example

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

SSN → Name, City

What the key?
{SSN, PhoneNumber}

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

16

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

17

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→C BC→A	or	A→BC B→AC

what are the keys here ?

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

18

Boyce-Codd Normal Form

A simple condition for removing anomalies from relations:

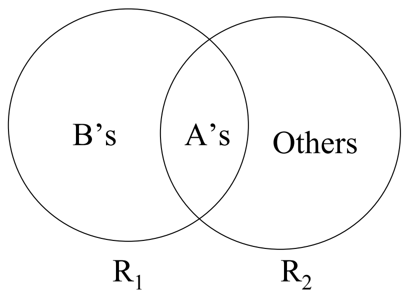
> A relation R is in BCNF if:
>
> If $A_1, ..., A_n \rightarrow B$ is a non-trivial dependency
>
> in R , then $\{A_1, ..., A_n\}$ is a superkey for R

In other words: there are no "bad" FDs

Equivalently:
$\forall$ X, either ($X^+ = X$) or (X^+ = 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 $R_2(A_1, \ldots, A_m$, [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	Seattle
Fred	123-45-6789	206-555-6543	Seattle
Joe	987-65-4321	908-555-2121	Westfield
Joe	987-65-4321	908-555-1234	Westfield

SSN → Name, City

What the key?

{SSN, PhoneNumber}

use SSN → Name, City to split

21

Example

Name	<u>SSN</u>	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

SSN → Name, City

<u>SSN</u>	<u>PhoneNumber</u>
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 ?

22

Example Decomposition

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

SSN → name, age

age → hairColor

Decompose in BCNF (in class):

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

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)

What are the keys ?

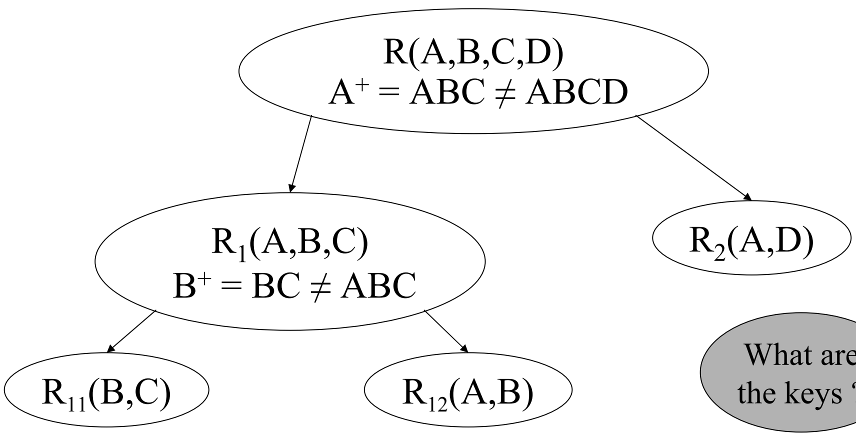
25

R(A,B,C,D)

Example

A → B

B → C



What are the keys ?

What happens if in R we first pick B^+ ? Or AB^+ ?

26

Decompositions in General

$R(A_1, ..., A_n, B_1, ..., B_m, C_1, ..., C_p)$

$R_1(A_1, ..., A_n, B_1, ..., B_m)$

$R_2(A_1, ..., A_n, C_1, ..., C_p)$

R_1 = projection of R on $A_1, ..., A_n, B_1, ..., B_m$
R_2 = projection of R on $A_1, ..., A_n, C_1, ..., C_p$

Theory of Decomposition

- Sometimes it is correct:

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

Name	Price
Gizmo	19.99
OneClick	24.99
~~Gizmo~~	~~19.99~~

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Lossless decomposition

Incorrect Decomposition

- Sometimes it is not:

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

What's incorrect ??

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Lossy decomposition

29

Decompositions in General

$R(A_1, ..., A_n, B_1, ..., B_m, C_1, ..., C_p)$

$R_1(A_1, ..., A_n, B_1, ..., B_m)$

$R_2(A_1, ..., A_n, C_1, ..., C_p)$

If $A_1, ..., A_n \rightarrow B_1, ..., B_m$
Then the decomposition is lossless

Note: don't need $A_1, ..., A_n \rightarrow C_1, ..., C_p$

BCNF decomposition is always lossless. WHY ?

30

3NF: A Problem with BCNF

Unit	Company	Product

Unit → Company
Company, Product → Unit

$Unit^+$ = Unit, Company

Unit	Company

Unit → Company

Unit	Product

We loose the FD: Company, Product → Unit !!

So What's the Problem?

Unit	Company
Galaga99	UW
Bingo	UW

Unit → Company

Unit	Product
Galaga99	Databases
Bingo	Databases

No problem so far. All *local* FD's are satisfied.
Let's put all the data back into a single table again:

Unit	Company	Product
Galaga99	UW	Databases
Bingo	UW	Databases

Violates the FD: Company, Product → Unit

The Problem

- We started with a table R and FD
- We decomposed R into BCNF tables R_1, R_2, … with their own FD_1, FD_2, …
- We can reconstruct R from R_1, R_2, …
- But we cannot reconstruct FD from FD_1, FD_2, …

33

Solution: 3rd Normal Form (3NF)

A simple condition for removing anomalies from relations:

A relation R is in 3rd normal form if :

Whenever there is a nontrivial dependency $A_1, A_2, ..., A_n \rightarrow B$ for R , then $\{A_1, A_2, ..., A_n\}$ a super-key for R,
or B is part of a key.

Tradeoff:

BCNF = no anomalies, but may lose some FDs
3NF = keeps all FDs, but may have some anomalies

34

3NF Decomposition Algorithm

3NF_Decompose(R)
let K = [all attributes that are part of some key]

find X s.t.: $X^+ - X - K \neq \emptyset$ and $X^+ \neq$ [all attributes]

if (not found) **then** "R is already in 3NF"

let $Y = X^+ - X - K$
let Z = [all attributes] - $(X \cup Y)$
decompose into $R1(X \cup Y)$ and $R2(X \cup Z)$
decompose, recursively, R1 and R2

Example of 3NF decomposition

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

$AB \to C$
$C \to D$
$D \to B$
$D \to E$

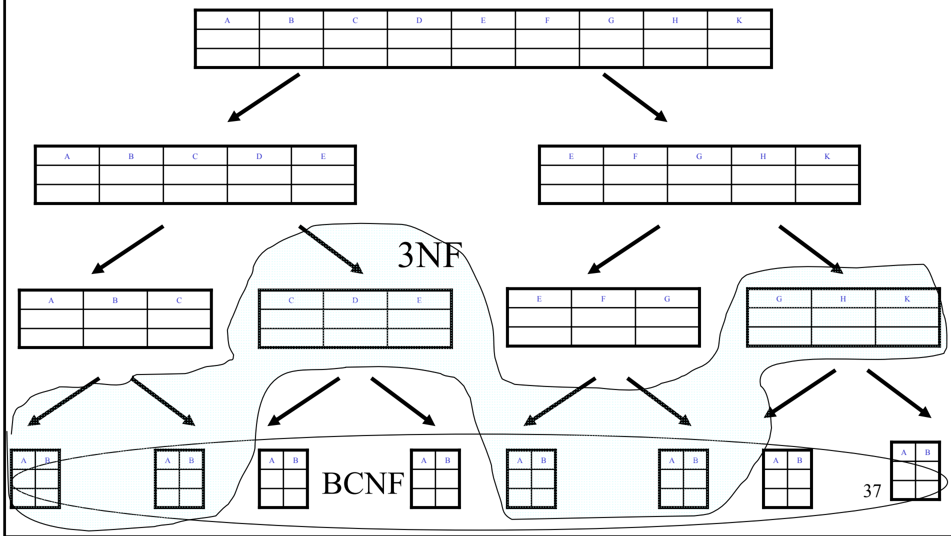
Keys: (need to compute X+, for several Xs)
AB, AC, AD

K = {A, B, C, D}

Pick X = C
C+ = BCDE
$C \to BDE$ is a BCNF violation
For 3NF: remove B, D (part of K):
$C \to E$ is a 3NF violation
Decompose: R1(C, E), R2(A,B,C,D)

R1 is in 3NF
R2 is in 3NF (because its keys: AB, AC, AD)

3NF v.s. BCNF Decomposition

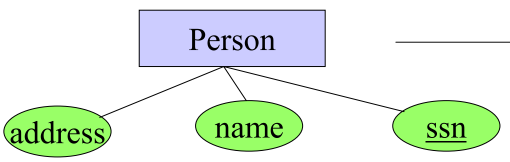


37

FD's for E/R Diagrams

Given a relation constructed from an E/R diagram, what is its key?

Rule 1: If the relation comes from an entity set,
the key of the relation is the set of attributes which is the key of the entity set.

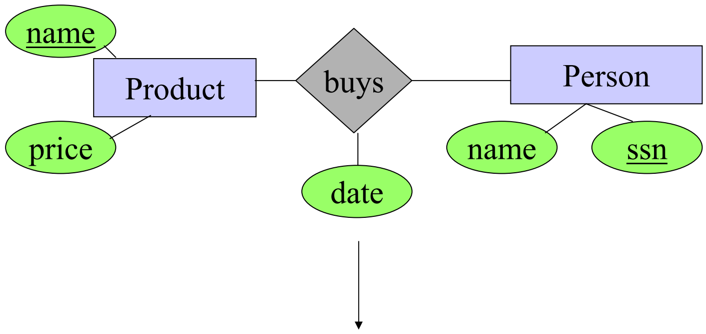


→ **Person(**address, name, ssn**)**

38

FD's for E/R Diagrams

Rule 2: If the relation comes from a many-many relationship, the key of the relation is the set of all attribute keys in the relations corresponding to the entity sets

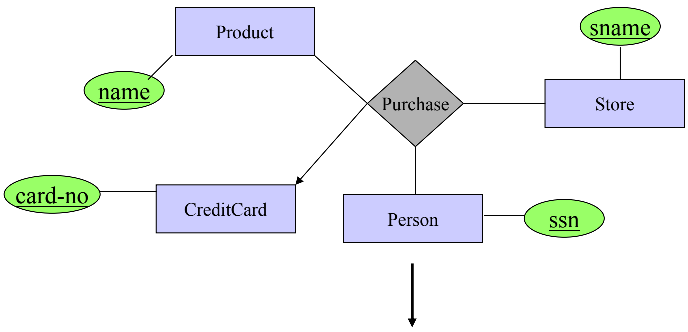


buys(<u>name, ssn</u>, date)

39

FD's for E/R Diagrams

Except: if there is an arrow from the relationship to E, then we don't need the key of E as part of the relation key.



Purchase(<u>name , sname, ssn</u>, card-no)

40

FD's for E/R Diagrams

More rules:

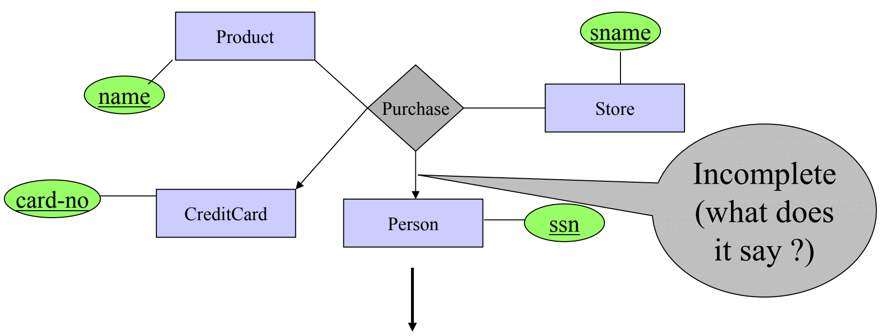
- Many-one, one-many, one-one relationships
- Multi-way relationships
- Weak entity sets

(Try to find them yourself, or check book)

41

FD's for E/R Diagrams

Say: "the CreditCard determines the Person"



Purchase(name , sname, ssn, card-no)

card-no → ssn

42