Lecture 09:

Friday, October 18, 2002

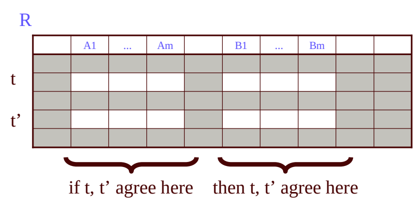
Outline

- Functional dependencies (3.4)
- Rules about FDs (3.5)
- Design of a Relational schema (3.6)

Functional Dependencies

Definition: $A_1, ..., A_m \rightarrow B_1, ..., B_n$ holds in R if:

$\forall t, t' \in R, (t.A_1=t'.A_1 \wedge ... \wedge t.A_m=t'.A_m \Rightarrow t.B_1=t'.B_1 \wedge ... \wedge t.B_m=t'.B_m)$



Formal definition of a key

- A **key** is a set of attributes $A_1, ..., A_n$ s.t. for any other attribute B, $A_1, ..., A_n \rightarrow B$
- A **minimal key** is a set of attributes which is a key and for which no subset is a key
- Note: book calls them **superkey** and **key**

Examples of Keys

- Product(name, price, category, color)
  - name, category → price
  - category → color

  Keys are: {name, category} and all supersets

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

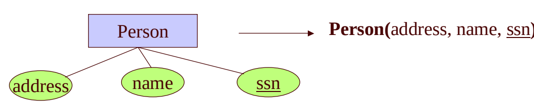
  Keys are: [in class]

Finding the Keys of a Relation

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

Rules:

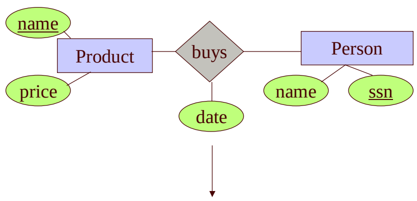
1. If the relation comes from an entity set,
   the key of the relation is the set of attributes which is the key of the entity set.



Finding the Keys

Rules:

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

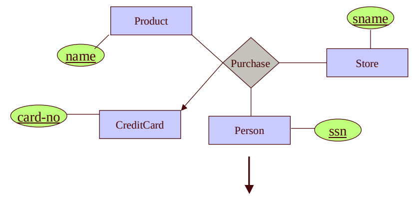


buys(<u>name, ssn</u>, date)

7

Finding the Keys

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.

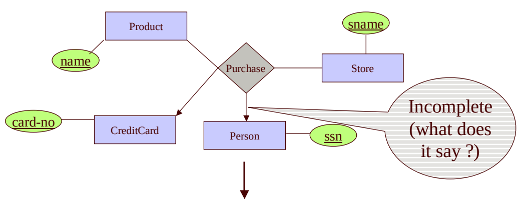


Purchase(<u>name , sname, ssn</u>, card-no)

8

Expressing Dependencies

Say: "the CreditCard determines the Person"



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

card-no → name

9

Finding the Keys

More rules in the book – please read !

10

Inference Rules for FD's

$A_1, A_2, \ldots A_n \rightarrow B_1, B_2, \ldots B_m$

Is equivalent to

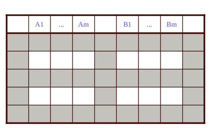
$A_1, A_2, \ldots A_n \rightarrow B_1$

$A_1, A_2, \ldots A_n \rightarrow B_2$

…

$A_1, A_2, \ldots A_n \rightarrow B_m$

Splitting rule and Combing rule



11

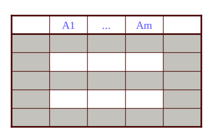
Inference Rules for FD's (continued)

$A_1, A_2, \ldots A_n \rightarrow A_i$

Trivial Rule

where i = 1, 2, ..., n

Why ?



12

Inference Rules for FD's (continued)

Transitive Closure Rule

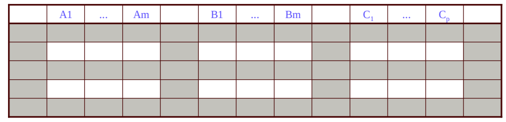
If $A_1, A_2, \ldots A_n \rightarrow B_1, B_2, \ldots, B_m$

and $B_1, B_2, \ldots B_m \rightarrow C_1, C_2, \ldots, C_p$

then $A_1, A_2, \ldots A_n \rightarrow C_1, C_2, \ldots, C_p$

Why ?

13



14

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

What else can we infer ? [in class]

15

Closure of a set of Attributes

Given a set of attributes $\{A1, \ldots, An\}$ and a set of dependencies S.
Problem: find all attributes *B* such that:
any relation which satisfies S also satisfies:
$A1, \ldots, An \rightarrow B$

The **closure** of $\{A1, \ldots, An\}$, denoted $\{A1, \ldots, An\}^+$,
is the set of all such attributes *B*

16

Closure Algorithm

Start with X={A1, …, An}.

Repeat until X doesn't change **do**:

if $B_1, B_2, \ldots B_n \rightarrow C$ is in S, **and**
$B_1, B_2, \ldots B_n$ are all in X, **and**
C is not in X

then

add C to X.

17

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

Example

A B → C
A D → E
B → D
A F → B

Closure of {A,B}: X = {A, B, }

Closure of {A, F}: X = {A, F, }

18

Why Is the Algorithm Correct ?

- Show the following by induction:
  - For every *B* in *X*:
    - *A1, …, An* → *B*
- Initially *X = {A1, …, An}* -- holds
- Induction step: *B1, …, Bm* in *X*
  - Implies *A1, …, An* → *B1, …, Bm*
  - We also have *B1, …, Bm* → *C*
  - By transitivity we have *A1, …, An* → *C*
- This shows that the algorithm is *sound*; need to show it is *complete*

19

Relational Schema Design (or Logical Design)

Main idea:

- Start with some relational schema
- Find out its FD’s
- Use them to design a better relational schema

20

Relational Schema Design (or Logical Design)

When a database is poorly designed we get anomalies:

- Redundancy: data is repeated
- Updated anomalies: need to change in several places
- Delete anomalies: may lose data when we don’t want

21

Relational Schema Design

Recall set attributes (persons with several phones):

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, but not SSN → PhoneNumber

Anomalies:

- Redundancy = repeat data
- Update anomalies = Fred moves to “Bellvue”
- Deletion anomalies = Fred drops all phone numbers: what is his city ?

22

Relation Decomposition

Break the relation into two:

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

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

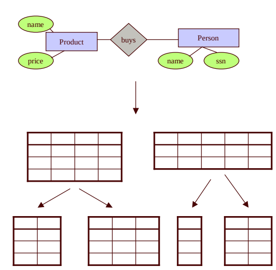
23

Relational Schema Design

Conceptual Model:

Relational Model: plus FD’s

Normalization: Eliminates anomalies



24

Decompositions in General

R(A_1, ..., A_n)

Create two relations R1(B1, ..., Bm) and R2(C1, ..., Cp)

such that: B1, ..., Bm ∪ C1, ..., Cp = A1, ..., An

and:

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

25

Incorrect Decomposition

- Sometimes it is incorrect:

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
DoubleClick	29.99	Camera

Decompose on : Name, Category and Price, Category

26

Incorrect Decomposition

Name	Category
Gizmo	Gadget
OneClick	Camera
DoubleClick	Camera

Price	Category
19.99	Gadget
24.99	Camera
29.99	Camera

When we put it back:

Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
OneClick	29.99	Camera
DoubleClick	24.99	Camera
DoubleClick	29.99	Camera

Cannot recover information

27

Normal Forms

First Normal Form = all attributes are atomic

Second Normal Form (2NF) = old and obsolete

Third Normal Form (3NF) = this lecture

Boyce Codd Normal Form (BCNF) = this lecture

Others...

28

Boyce-Codd Normal Form

A simple condition for removing anomalies from relations:

A relation R is in BCNF if:

Whenever there is a nontrivial dependency $A_1, ..., A_n \rightarrow B$ in R , $\{A_1, ..., A_n\}$ is a key for R

In English (though a bit vague):

Whenever a set of attributes of *R* is determining another attribute, should determine ***all*** the attributes of *R*.

29

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

What are the dependencies?
SSN → Name, City
What are the keys?
{Name, SSN, PhoneNumber}
Is it in BCNF?

30

Decompose it into BCNF

Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

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

31

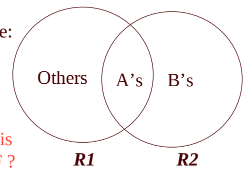
Summary of BCNF Decomposition

Find a dependency that violates the BCNF condition:

$A_1, A_2, \ldots A_n \longrightarrow B_1, B_2, \ldots B_m$

Heuristics: choose $B_1, B_2, \ldots B_m$ "as large as possible"

Decompose:



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

Continue until there are no BCNF violations left.

32

Example Decomposition

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

SSN → name, age

age → hairColor

Decompose in BCNF (in class):

Step 1: find all keys

Step 2: now decompose

33

Other Example

- R(A,B,C,D) A → B, B → C
- Key: A, D
- Violations of BCNF: A → B, A → C, A → BC
- Pick A → BC: split into R1(A,BC) R2(A,D)
- What happens if we pick A → B first ?

34

Correct Decompositions

A decomposition is *lossless* if we can recover:

R(A,B,C)

↓ Decompose

R1(A,B) R2(A,C)

↓ Recover

R'(A,B,C) should be the same as R(A,B,C)

R' is in general larger than R. Must ensure R' = R

35

Correct Decompositions

- Given R(A,B,C) s.t. A → B, the decomposition into R1(A,B), R2(A,C) is lossless

36

3NF: A Problem with BCNF

Unit	Company	Product

FD's: Unit → Company; Company, Product → Unit

So, there is a BCNF violation, and we decompose.

Unit	Company

Unit → Company

Unit	Product

No FDs

So What's the Problem?

Unit	Company
Galaga99	UW
Bingo	UW

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 dependency: company, product -> unit!

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.