Lecture 8:
Database Design

Monday, January 23, 2006

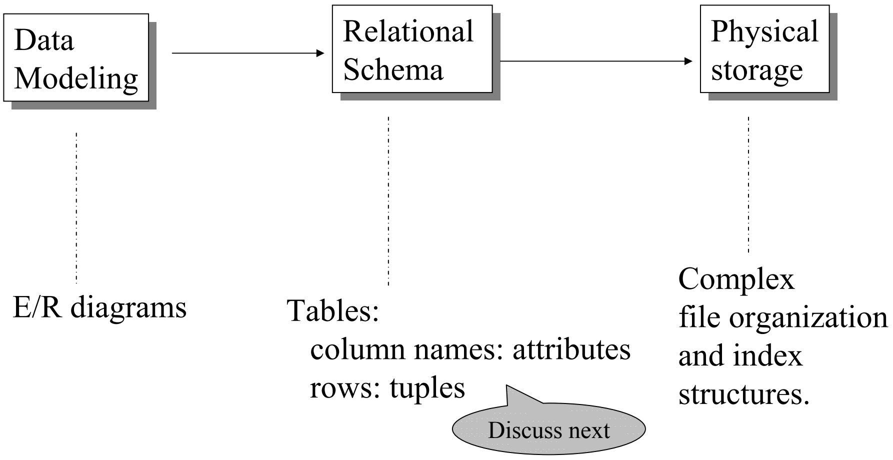
Announcements/Reminders

- Homework 1: solutions are posted
- Homework 2: posted (due Wed. Feb.1st)
- Project Phase 1 due Wednesday

Outline

- The relational data model: 3.1
- Functional dependencies: 3.4
  - Will continue next time

The Relational Data Model



Data Modeling

Relational Schema

Physical storage

E/R diagrams

Tables:
column names: attributes
rows: tuples

Complex
file organization
and index
structures.

Discuss next

Normal Forms

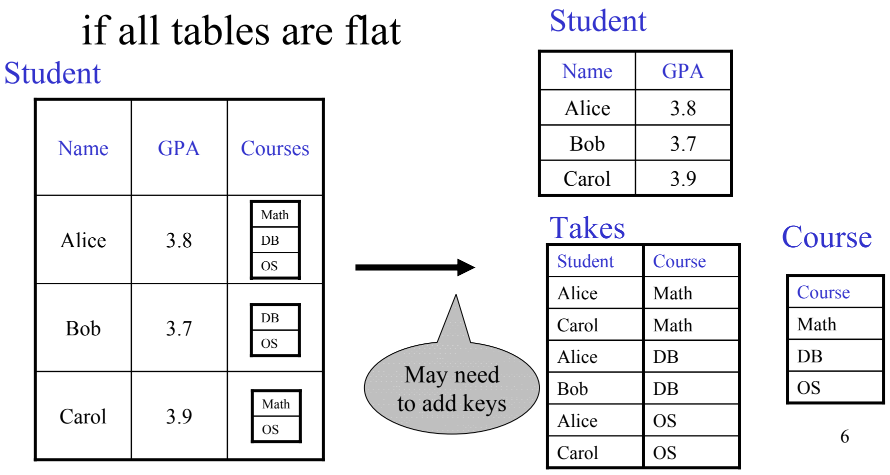
Idea: replace one relational schema with another one, which is *better*.

Hence, *normal form,* NF.

- 1NF = rather trivial
- 3NF, BCNF = next time
- Other normal forms = in the book

First Normal Form (1NF)

- A database schema is in First Normal Form if all tables are flat



Student

Name	GPA	Courses
Alice	3.8	Math DB OS
Bob	3.7	DB OS
Carol	3.9	Math OS

May need to add keys

Student

Name	GPA
Alice	3.8
Bob	3.7
Carol	3.9

Takes

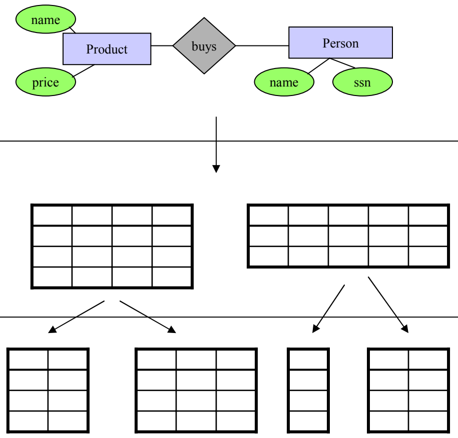
Student	Course
Alice	Math
Carol	Math
Alice	DB
Bob	DB
Alice	OS
Carol	OS

Course

Course
Math
DB
OS

Relational Schema Design

Conceptual Model:



Relational Model:
plus FD's

Normalization:
Eliminates ***anomalies***

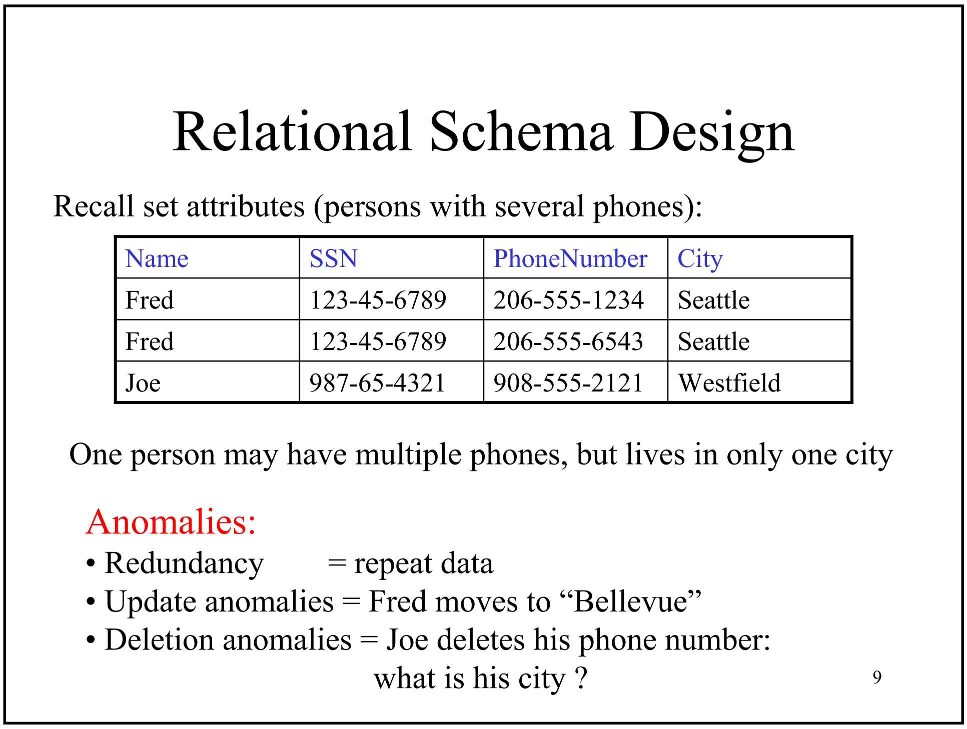
Data Anomalies

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



Relational Schema Design

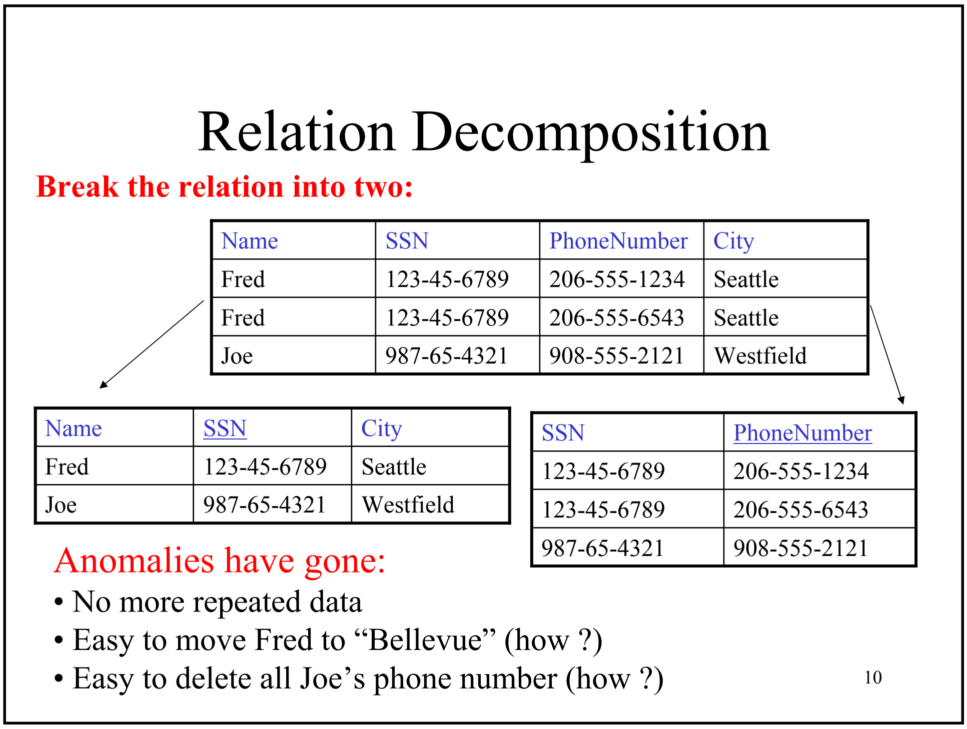
Recall set attributes (persons with several phones):

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

One person may have multiple phones, but lives in only one city

Anomalies:

- Redundancy = repeat data
- Update anomalies = Fred moves to “Bellevue”
- Deletion anomalies = Joe deletes his phone number: what is his city ?



Relation Decomposition

Break the relation into two:

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

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

Anomalies have gone:

- No more repeated data
- Easy to move Fred to “Bellevue” (how ?)
- Easy to delete all Joe’s phone number (how ?)

Relational Schema Design (or Logical Design)

Main idea:

- Start with some relational schema
- Find out its ***functional dependencies***
- Use them to design a better relational schema

Functional Dependencies

- A form of constraint
  - hence, part of the schema
- Finding them is part of the database design
- Also used in normalizing the relations

Functional Dependencies

Definition:

If two tuples agree on the attributes

$\boxed{A_1, A_2, \ldots, A_n}$

then they must also agree on the attributes

$\boxed{B_1, B_2, \ldots, B_m}$

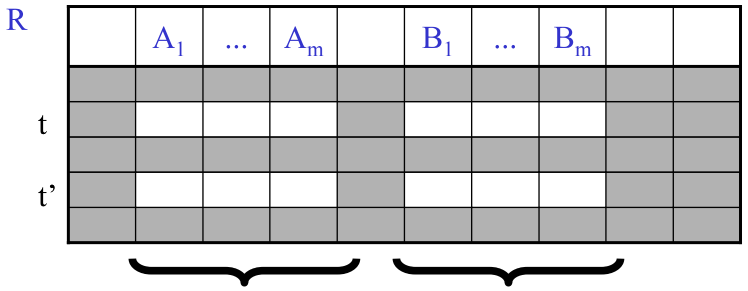
Formally:

$\boxed{A_1, A_2, \ldots, A_n \rightarrow B_1, B_2, \ldots, B_m}$

When Does an FD Hold

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

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



if t, t' agree here then t, t' agree here

Examples

An FD holds, or does not hold on an instance:

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

EmpID → Name, Phone, Position

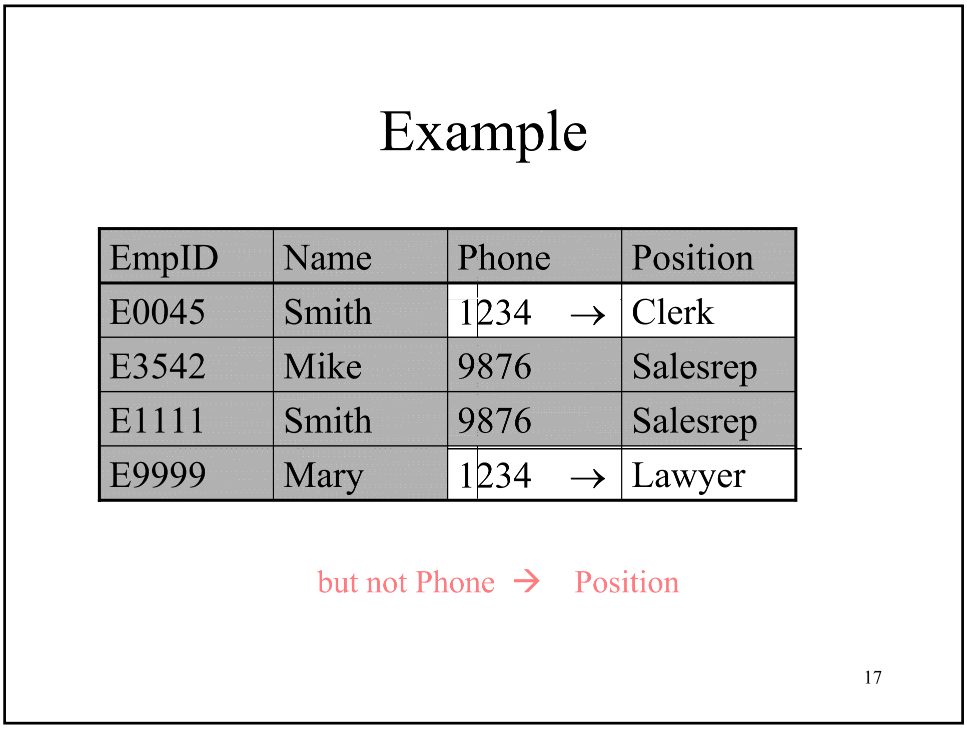
Position → Phone

but not Phone → Position

Example

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E3542	Mike	9876 ←	Salesrep
E1111	Smith	9876 ←	Salesrep
E9999	Mary	1234	Lawyer

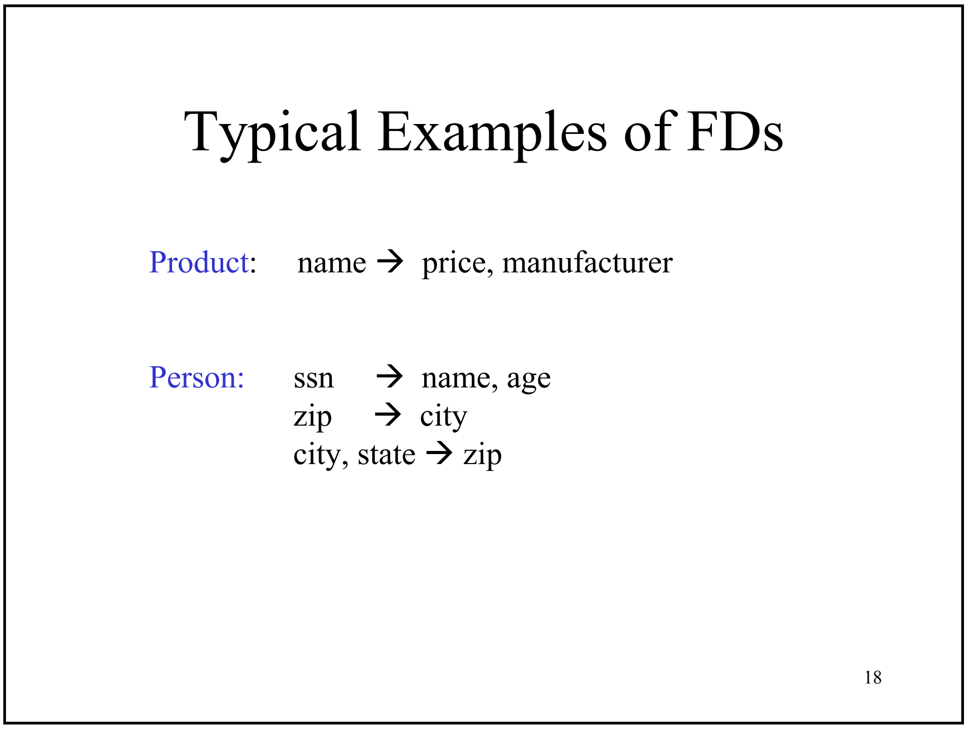
Position → Phone



Example

EmpID	Name	Phone	Position
E0045	Smith	1234 →	Clerk
E3542	Mike	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234 →	Lawyer

but not Phone → Position



Typical Examples of FDs

Product: name → price, manufacturer

Person: ssn → name, age
zip → city
city, state → zip

Example

FD’s are constraints:
- On some instances they hold
- On others they don’t

name → color
category → department
color, category → price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Green	Toys	99

Does this instance satisfy all the FDs ?

Example

name → color
category → department
color, category → price

name	category	color	department	price
Gizmo	Gadget	Green	Toys	49
Tweaker	Gadget	Black	Toys	99
Gizmo	Stationary	Green	Office-supp.	59

What about this one ?

An Interesting Observation

If all these FDs are true:

name → color
category → department
color, category → price

Then this FD also holds:

name, category → price

Why ??

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

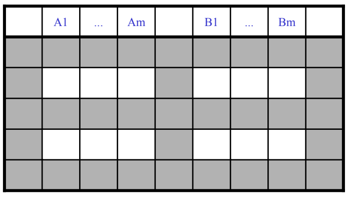
Inference Rules for FD’s

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

Splitting rule and Combing rule

Is equivalent to

$$\begin{gathered} A_1, A_2, \ldots, A_n \rightarrow B_1 \\ A_1, A_2, \ldots, A_n \rightarrow B_2 \\ \ldots\ldots \\ A_1, A_2, \ldots, A_n \rightarrow B_m \end{gathered}$$



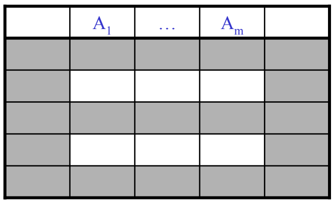
Inference Rules for FD’s (continued)

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

Trivial Rule

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

Why ?



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 ?

	A_1	…	A_m		B_1	…	B_m		C_1	…	C_p	
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
▨				▨				▨				▨
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨
▨				▨				▨				▨
▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨	▨

Example (continued)

Start from the following FDs:

1. name → color
2. category → department
3. color, category → price

Infer the following FDs:

Inferred FD	Which Rule did we apply ?
4. name, category → name	
5. name, category → color	
6. name, category → category	
7. name, category → color, category	
8. name, category → price	

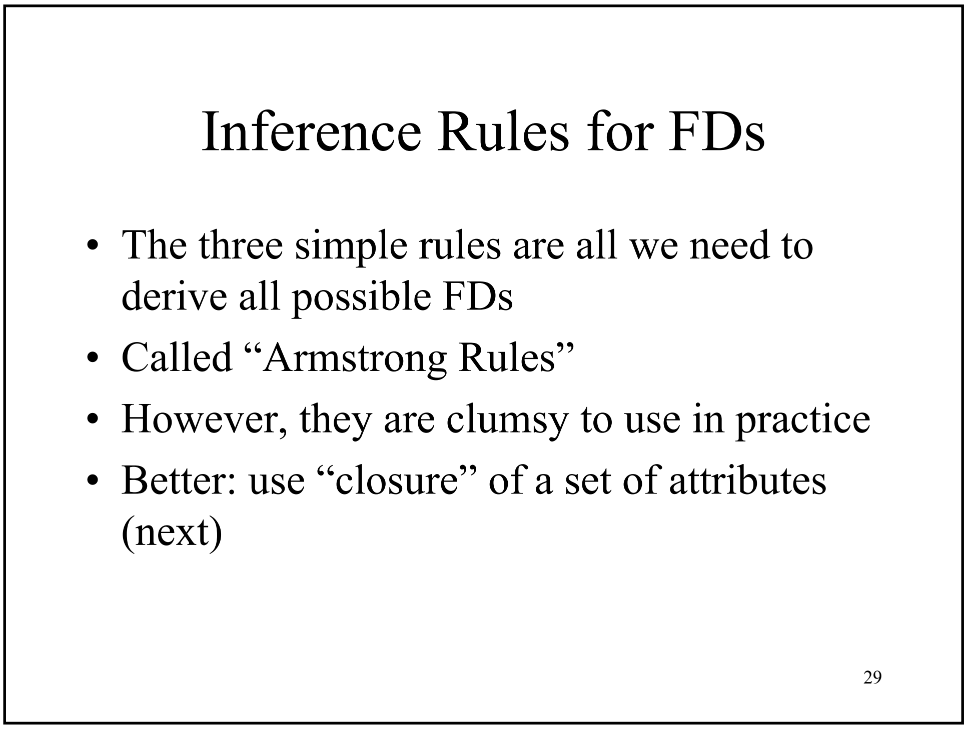
Example (continued)

Answers:

1. name → color
2. category → department
3. color, category → price

Inferred FD	Which Rule did we apply ?
4. name, category → name	Trivial rule
5. name, category → color	Transitivity on 4, 1
6. name, category → category	Trivial rule
7. name, category → color, category	Split/combine on 5, 6
8. name, category → price	Transitivity on 3, 7

THIS IS TOO HARD ! Let’s see an easier way.



Inference Rules for FDs

- The three simple rules are all we need to derive all possible FDs
- Called “Armstrong Rules”
- However, they are clumsy to use in practice
- Better: use “closure” of a set of attributes (next)