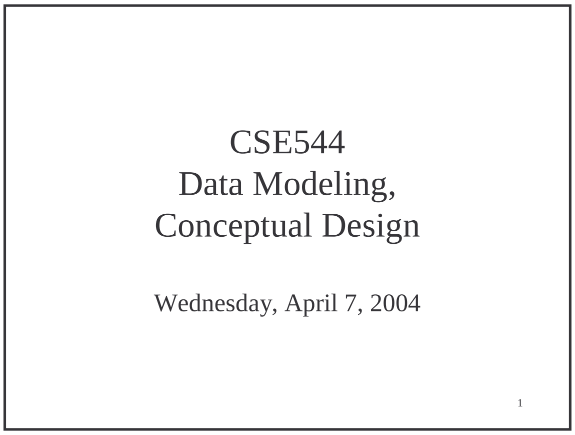
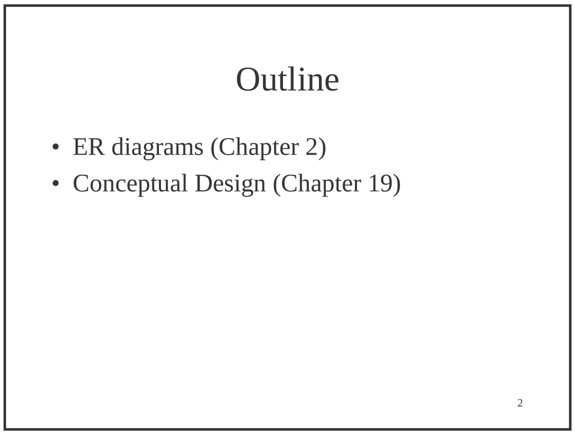


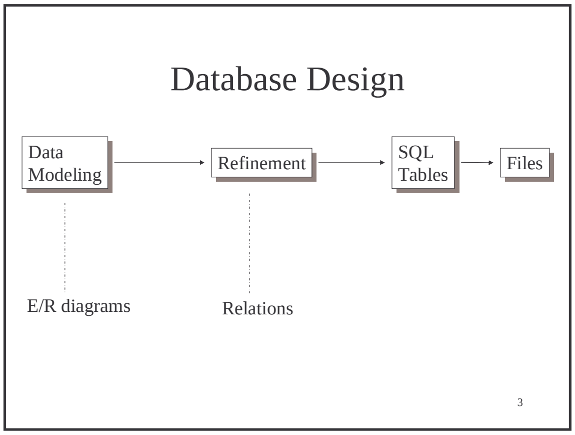
CSE544
Data Modeling,
Conceptual Design

Wednesday, April 7, 2004



Outline

- ER diagrams (Chapter 2)
- Conceptual Design (Chapter 19)

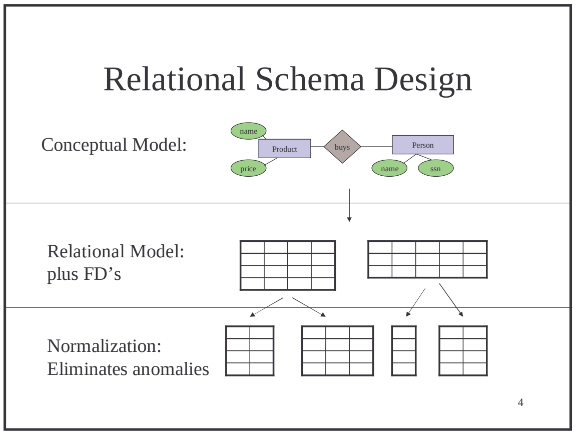


Database Design

Data Modeling → Refinement → SQL Tables → Files

E/R diagrams

Relations

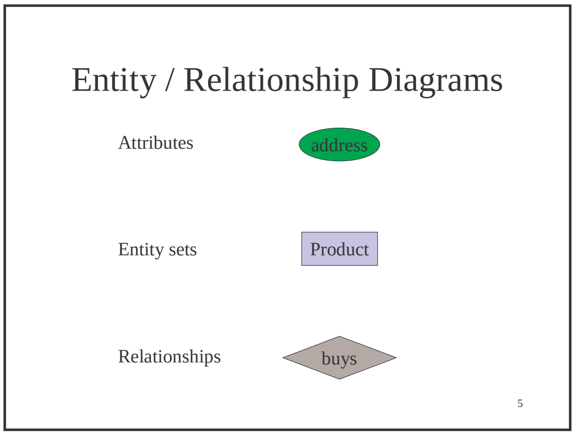


Relational Schema Design

Conceptual Model:

Relational Model:
plus FD’s

Normalization:
Eliminates anomalies

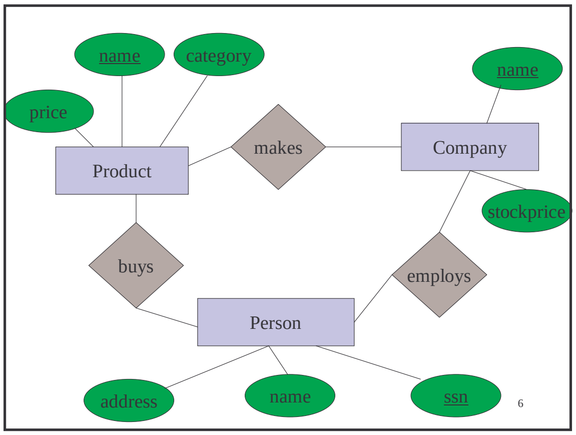


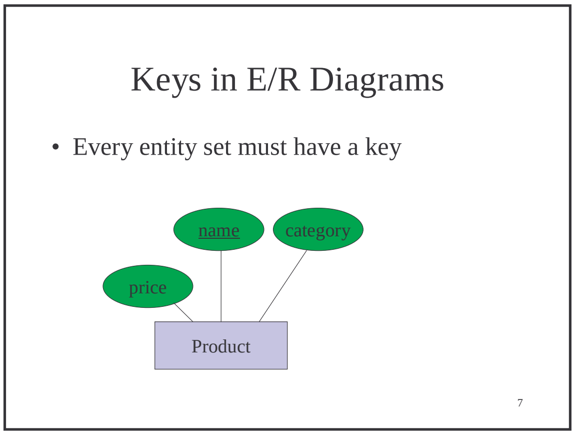
Entity / Relationship Diagrams

Attributes

Entity sets

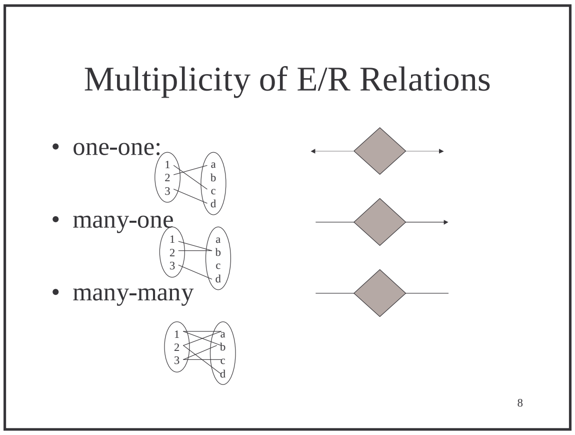
Relationships





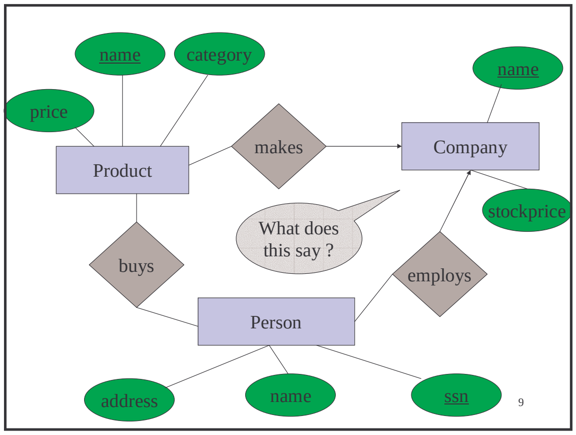
Keys in E/R Diagrams

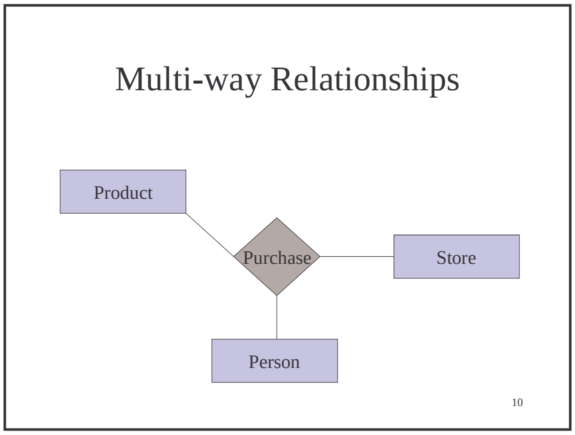
- Every entity set must have a key



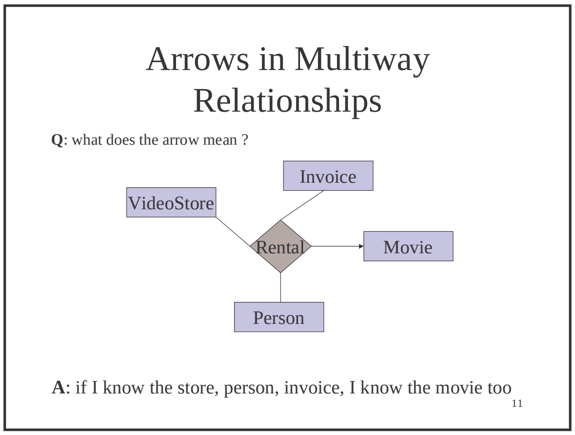
Multiplicity of E/R Relations

- one-one:
- many-one
- many-many





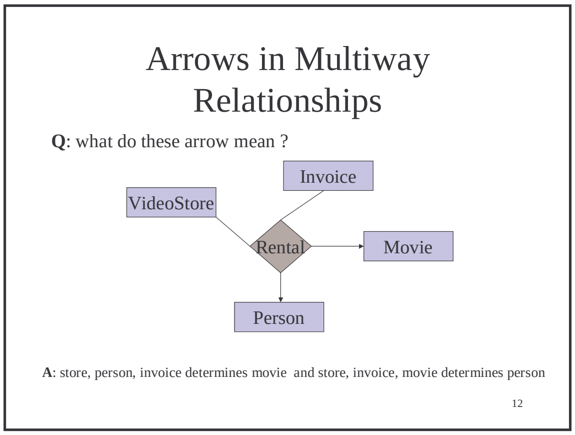
Multi-way Relationships



Arrows in Multiway Relationships

Q: what does the arrow mean ?

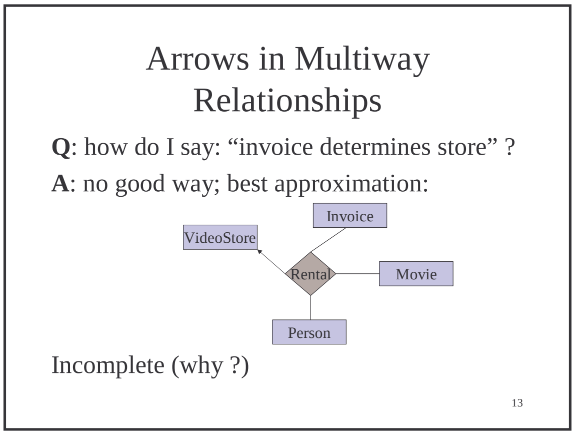
A: if I know the store, person, invoice, I know the movie too



Arrows in Multiway Relationships

Q: what do these arrow mean ?

A: store, person, invoice determines movie and store, invoice, movie determines person

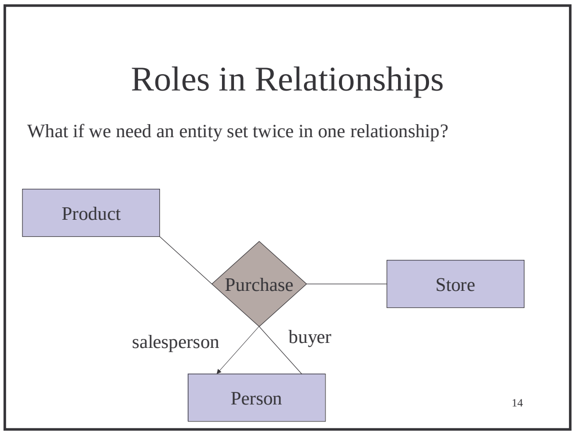


Arrows in Multiway Relationships

Q: how do I say: “invoice determines store” ?

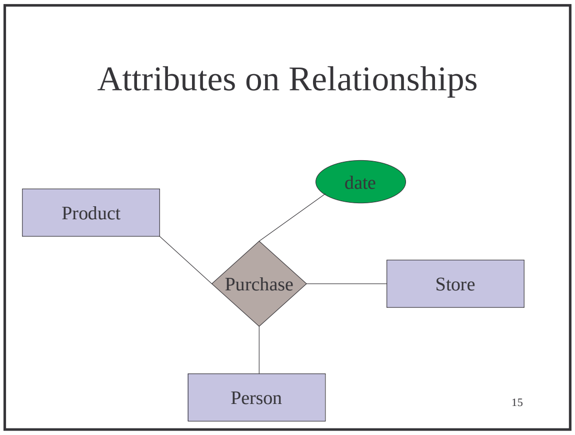
A: no good way; best approximation:

Incomplete (why ?)

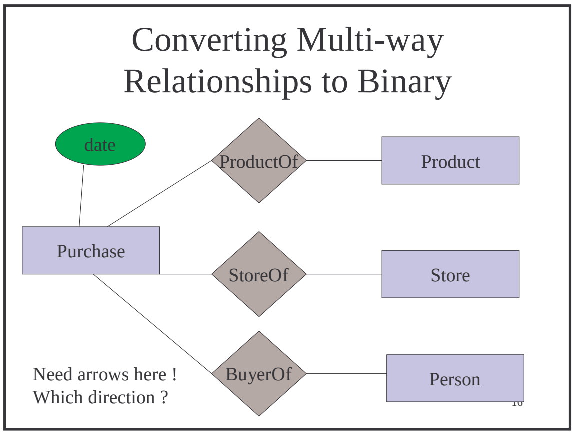


Roles in Relationships

What if we need an entity set twice in one relationship?

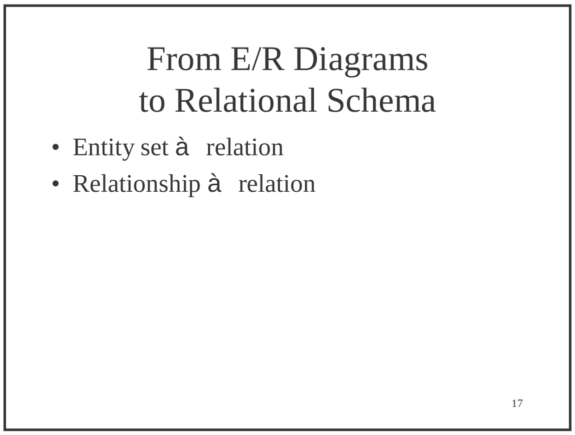


Attributes on Relationships



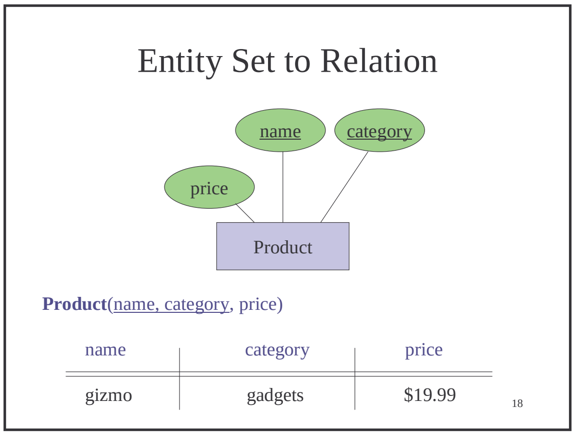
Converting Multi-way Relationships to Binary

Need arrows here !
Which direction ?



From E/R Diagrams to Relational Schema

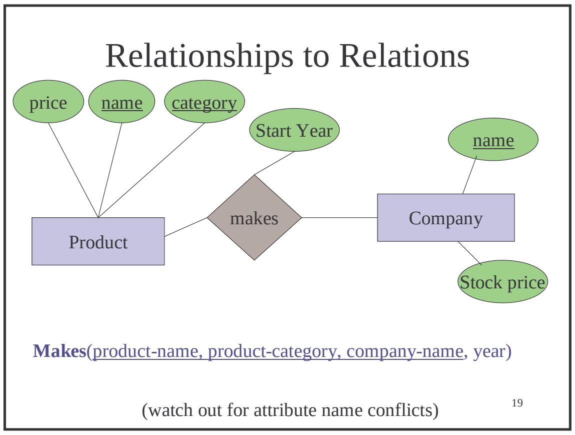
- Entity set à relation
- Relationship à relation



Entity Set to Relation

Product(name, category, price)

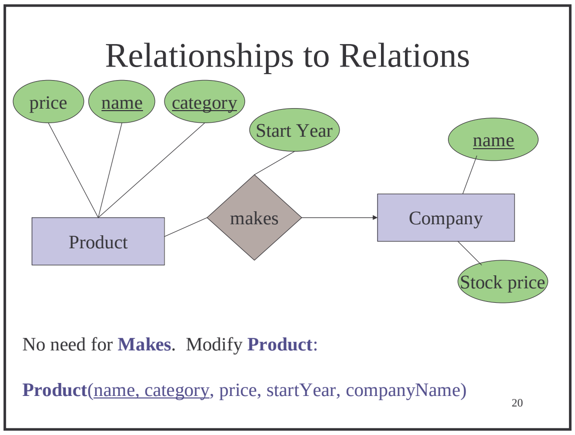
name	category	price
gizmo	gadgets	$19.99



Relationships to Relations

Makes(product-name, product-category, company-name, year)

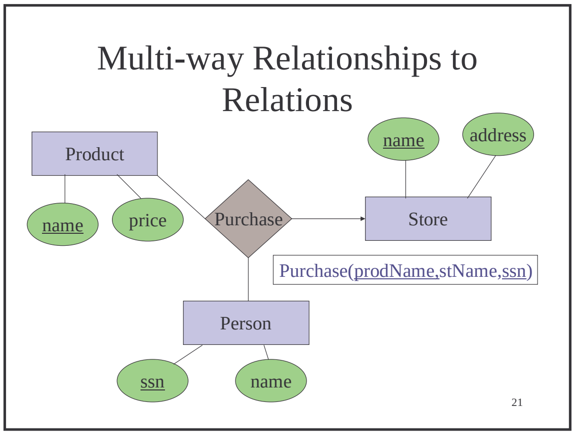
(watch out for attribute name conflicts)



Relationships to Relations

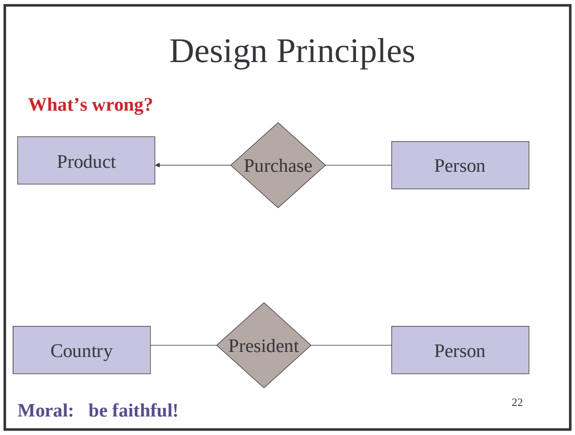
No need for **Makes**. Modify **Product**:

Product(name, category, price, startYear, companyName)



Multi-way Relationships to Relations

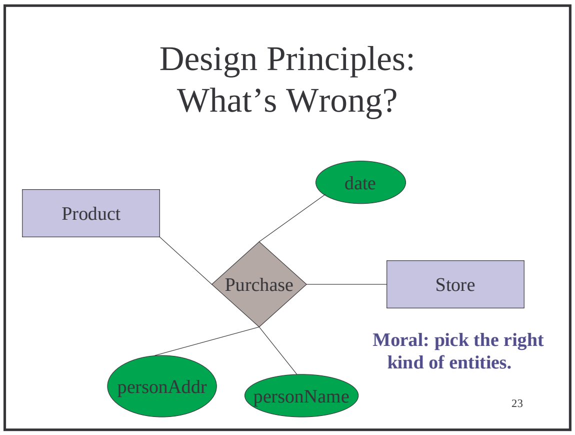
Purchase(prodName,stName,ssn)



Design Principles

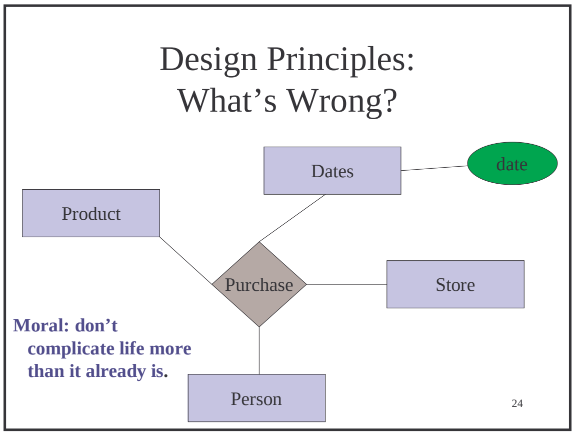
What's wrong?

Moral: be faithful!



Design Principles: What's Wrong?

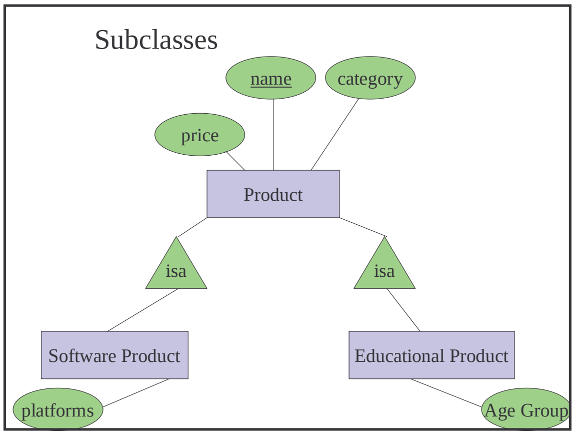
Moral: pick the right kind of entities.



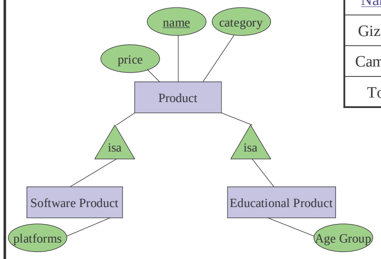
Design Principles: What's Wrong?

Moral: don't complicate life more than it already is.

Subclasses



Subclasses to Relations



Product

Name	Price	Category
Gizmo	99	gadget
Camera	49	photo
Toy	39	gadget

Sw.Product

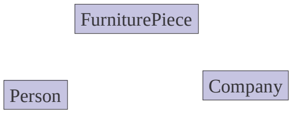
Name	platforms
Gizmo	unix

Ed.Product

Name	Age Group
Gizmo	todler
Toy	retired

Notice: subclass = subset
Alternative: disjoint classes (Java, C++)

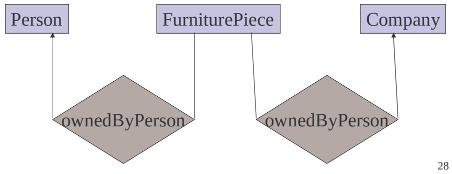
Modeling UnionTypes With Subclasses



Each piece of furniture is owned either by a person, or by a company

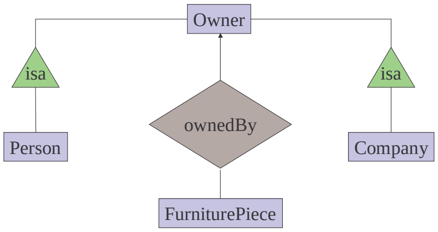
Modeling Union Types with Subclasses

Solution 1. Acceptable, imperfect (What’s wrong ?)

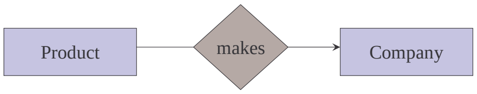


Modeling Union Types with Subclasses

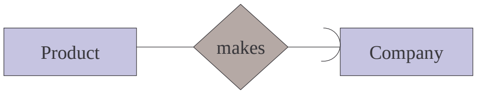
Solution 2: better



Referential Integrity Constraints

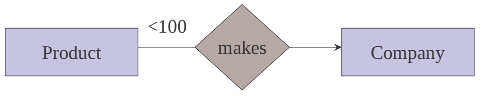


Each product made by at most one company.
Some products made by no company



Each product made by *exactly* one company.

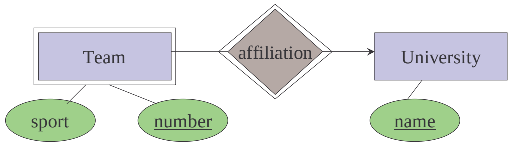
Other Constraints



What does this mean ?

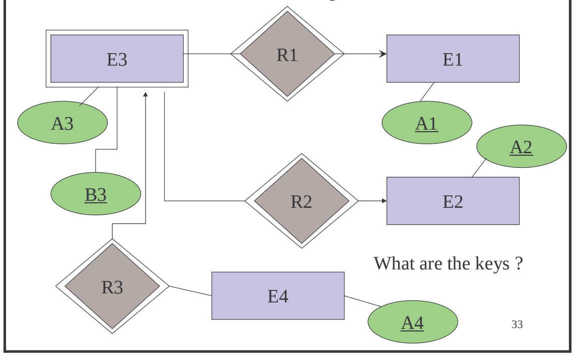
Weak Entity Sets

Entity sets are weak when their key comes from other classes to which they are related.



University(name)
Team(universityName, number, sport)

Weak Entity Sets



What are the keys ?

Schema Refinement

- For the relational model
- Relation: $R(A_1, A_2, \ldots, A_m)$
  - Schema: relation name, attribute names
  - Instance: a mathematical m-ary relation
- Database: $R_1, R_2, \ldots, R_n$
  - Schema
  - Instance
- Schema refinement = *normalization*

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

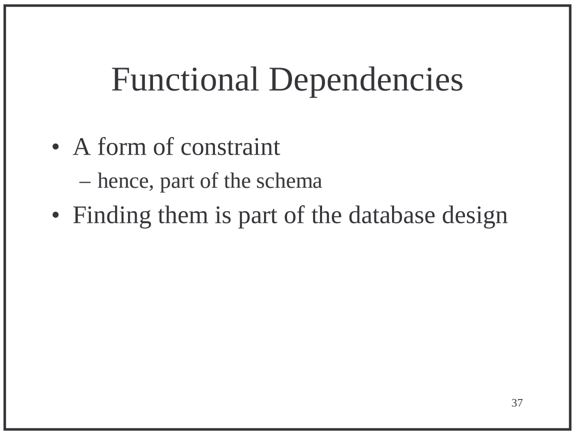
Course

Course
Math
DB
OS

More Normal Forms

- Based on Functional Dependencies
  - 2nd Normal Form (obsolete)
  - 3rd Normal Form
  - Boyce Codd Normal Form (BCNF)
- Based on Multivalued Dependencies
  - 4th Normal Form
- Based on Join Dependencies
  - 5th Normal Form

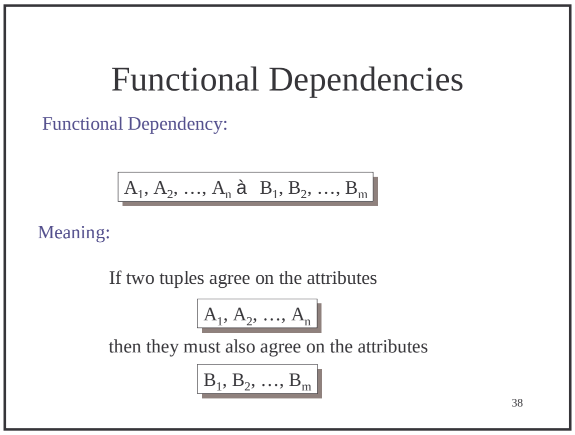
Discuss next



Functional Dependencies

- A form of constraint
  - – hence, part of the schema
- Finding them is part of the database design

37



Functional Dependencies

Functional Dependency:

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

Meaning:

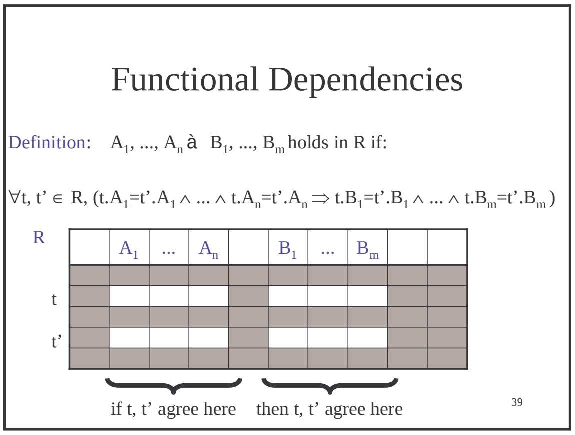
If two tuples agree on the attributes

$A_1, A_2, \ldots, A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots, B_m$

38



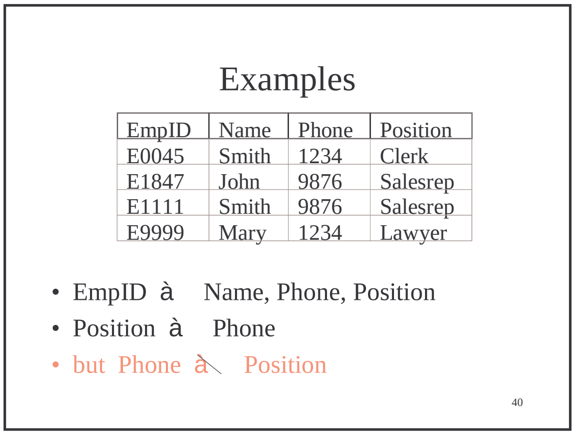
Functional Dependencies

Definition: $A_1, \ldots, A_n$ à $B_1, \ldots, B_m$ holds in R if:

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

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

39

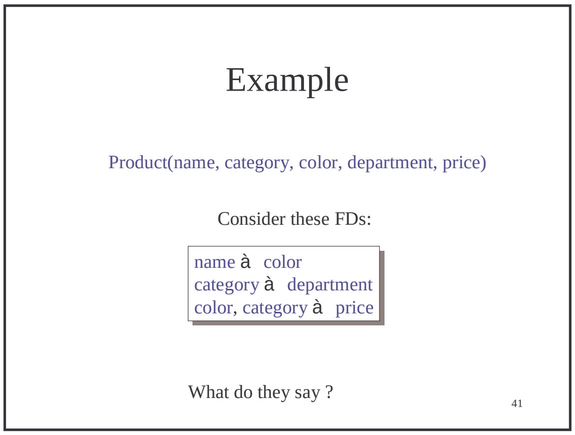


Examples

EmpID	Name	Phone	Position
E0045	Smith	1234	Clerk
E1847	John	9876	Salesrep
E1111	Smith	9876	Salesrep
E9999	Mary	1234	Lawyer

- EmpID à Name, Phone, Position
- Position à Phone
- but Phone ~~à~~ Position

40



Example

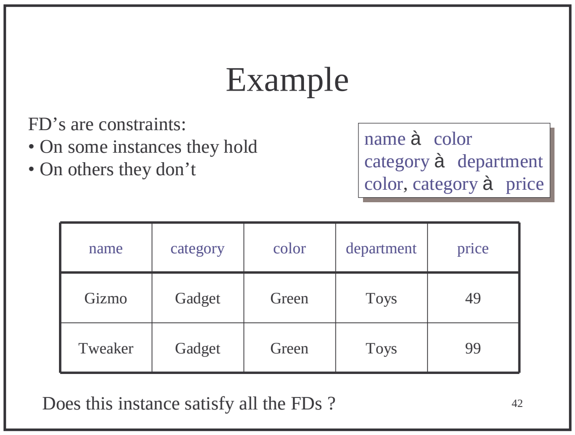
Product(name, category, color, department, price)

Consider these FDs:

name à color
category à department
color, category à price

What do they say ?

41



Example

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

name à color
category à department
color, category à price

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

Does this instance satisfy all the FDs ?

42

Example

name à color
category à department
color, category à 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 ?

Inference

If some FDs are satisfied, then
others are satisfied too

If all these FDs are true:

name à color
category à department
color, category à price

Then this FD also holds:

name, category à price

Why ??

We say that the new FD is *implied*

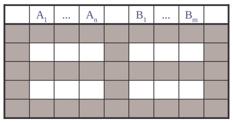
Armstrong's Axioms

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

Splitting rule
and
Combing rule

Is equivalent to

$A_1, A_2, \ldots, A_n$ à B_1
$A_1, A_2, \ldots, A_n$ à B_2
. . . . .
$A_1, A_2, \ldots, A_n$ à B_m



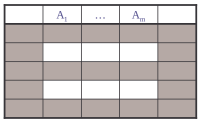
Armstrong's Axioms

$A_1, A_2, \ldots, A_n$ à A_i

Trivial Rule

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

Why ?



Armstrong's Axioms

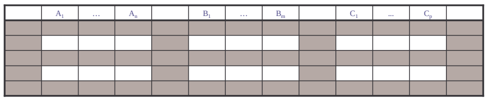
Transitive Closure Rule

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

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

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

Why ?



Example (continued)

From:

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

To: name, category à price

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

Example (continued)

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

Answers:

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

Closure of a Set of FDs

Definition. Given a set F of functional dependencies, the *closure*, F^+, denotes all FDs *implied* by F

Theorem. Armstrong axioms are *sound* and *complete* for computing F^+

What do *sound* and *complete* mean ?

Variation

Augmentation

If $A_1, A_2, \dots, A_n$ à B

then $A_1, A_2, \dots, A_n, C_1, C_2, \dots, C_p$ à B

Augmentation follows from trivial rules and transitivity
How ?

Problem: Compute F^+

Given F compute its closure F^+.

How to proceed ?

- Apply Armstrong's Axioms repeatedly
- Better: use the *Closure Algorithm* for a set of attributes (next)

Closure of a set of Attributes

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

The **closure**, $\{A_1, \dots, A_n\}^+$, is the set of attributes B s.t. $A_1, \dots, A_n$ à B

Example:

name à color
category à department
color, category à price

Closures:

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

Closure Algorithm (for Attributes)

```
Start with X={A1, …, An}.

Repeat until X doesn't change do:

    if   B_1, …, B_n à C  is a FD and
         B_1, …, B_n are all in X
    then add C to X.
```

Example:

name à color
category à department
color, category à price

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

Closure Algorithm (for 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 $X \rightarrow Y$, s.t. $Y \subseteq X^+$ and $X \cap Y = \emptyset$:

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

Keys

- A **superkey** is a set of attributes $A_1, ..., A_n$ s.t. $A_1, ..., A_n \rightarrow B$ for all attributes B
- A **key** is a minimal superkey

Computing Keys

- Compute X^+ for all sets X
- If X^+ = all attributes, then X is a superkey
- Consider only the minimal superkeys

Note: there can be exponentially many keys !

- Example: R(A,B,C), AB→ C, BC→ A
  Keys: AB and BC

Examples of Keys

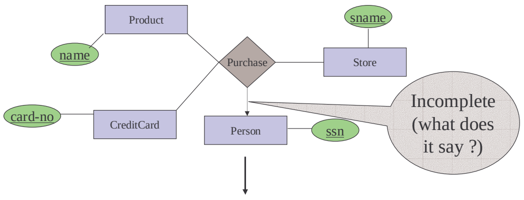
- Product(name, price, category, color)
  name, category → price
  category → color

  Key: {name, category} Superkeys: supersets
- Enrollment(student, address, course, room, time)
  student → address
  room, time → course
  student, course → room, time

  Keys are: [in class]

FD's for E/R Diagrams

Say: "the CreditCard determines the Person"



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

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

Schema refinement means removing the data anomalies.

61

Data Anomalies

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

SSN à Name, City   but not SSN à PhoneNumber

Anomalies:

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

62

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

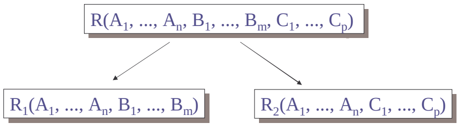
SSN	PhoneNumber
123-45-6789	206-555-1234
123-45-6789	206-555-6543
987-65-4321	908-555-2121

Anomalies have gone:

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

63

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 $A_1, ..., A_n, C_1, ..., C_p$

64

Problems With Decomposition

- Can we get the data back correctly ?
  - Lossless decomposition
  - Discuss next
- Can we recover the FD's on the 'big' table from the FD's on the small tables ?
  - Dependency-preserving decomposition
  - Figure out yourself, or read 19.5.2

65

Lossless 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

66

Lossy Decomposition

- Sometimes it is not:

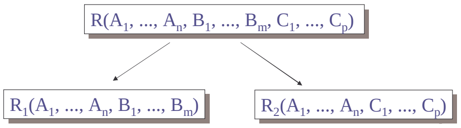
Name	Price	Category
Gizmo	19.99	Gadget
OneClick	24.99	Camera
Gizmo	19.99	Camera

What's wrong ??

Name	Category
Gizmo	Gadget
OneClick	Camera
Gizmo	Camera

Price	Category
19.99	Gadget
24.99	Camera
19.99	Camera

Decompositions in General



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

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

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

Theorem If $A_1, ..., A_n$ à $B_1, ..., B_m$
Then the decomposition is lossless

Note: don't need necessarily $A_1, ..., A_n$ à $C_1, ..., C_p$

Example: name à price, hence the first decomposition is lossless

Normal Forms

- Decomposition into Boyce Codd Normal Form (BCNF)
  - Losselss
- Decomposition into 3rd Normal Form
  - Losless
  - Dependency preserving

Boyce-Codd Normal Form

A simple condition for removing anomalies from relations:

A relation R is in BCNF if:

If $A_1, ..., A_n$ à B is a non-trivial dependency

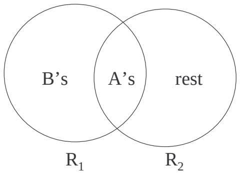
in R , then $\{A_1, ..., A_n\}$ is a superkey for R

Equivalently: for any set of attributes X,
either $X^+ = X$
or X^+ = all attributes

BCNF Decomposition Algorithm

```
Repeat
    choose $A_1, …, A_m$ à $B_1, …, B_n$ that violates the BNCF condition
    split R into $R_1(A_1, …, A_m, B_1, …, B_n)$ and $R_2(A_1, …, A_m, [rest])$
    continue with both $R_1$ and $R_2$
Until no more violations
```

Heuristics:
choose $B_1, …, B_n$
"as large as possible"



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

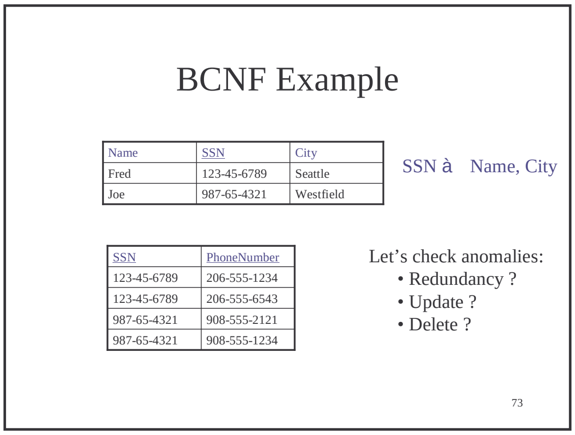
Note: need to compute the FDs on R_1, R_2 (how ?)

BCNF 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

FD: SSN à Name, City
Key: {SSN, PhoneNumber}
Is it in BCNF?

Another way: SSN^+ ={SSN, Name, City} but no PhoneNumber



BCNF Example

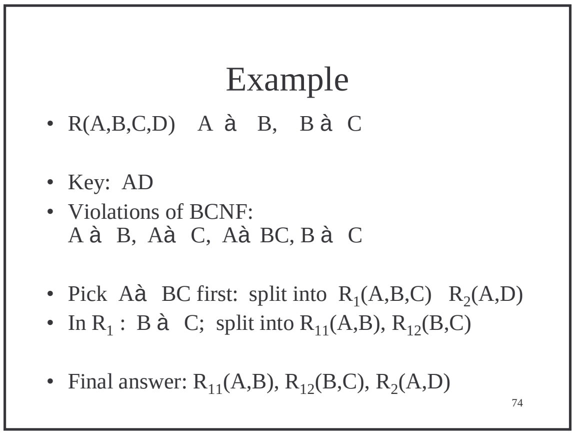
Name	SSN	City
Fred	123-45-6789	Seattle
Joe	987-65-4321	Westfield

SSN à 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

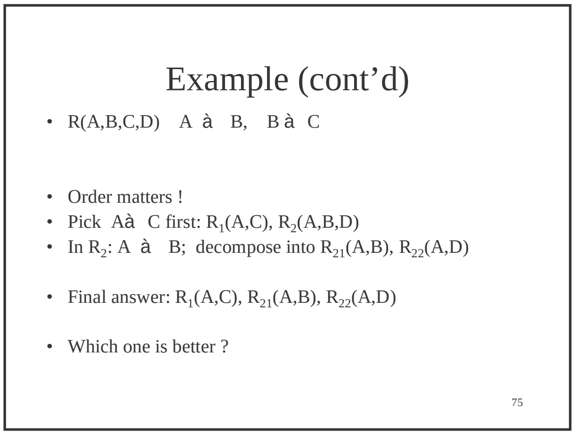
Let's check anomalies:

- Redundancy ?
- Update ?
- Delete ?



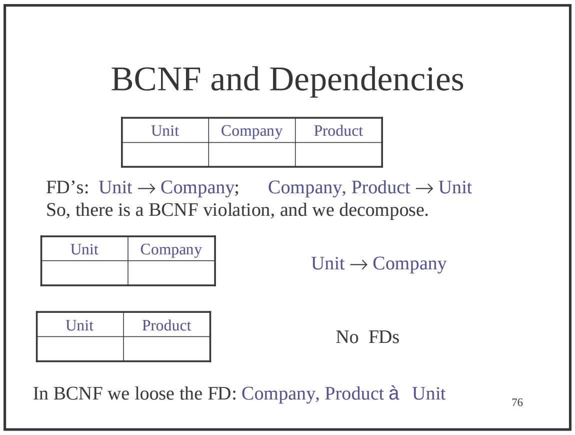
Example

- R(A,B,C,D) A à B, B à C
- Key: AD
- Violations of BCNF:
  A à B, Aà C, Aà BC, B à C
- Pick Aà BC first: split into R_1(A,B,C) R_2(A,D)
- In R_1 : B à C; split into R_{11}(A,B), R_{12}(B,C)
- Final answer: R_{11}(A,B), R_{12}(B,C), R_2(A,D)



Example (cont'd)

- R(A,B,C,D) A à B, B à C
- Order matters !
- Pick Aà C first: R_1(A,C), R_2(A,B,D)
- In R_2: A à B; decompose into R_{21}(A,B), R_{22}(A,D)
- Final answer: R_1(A,C), R_{21}(A,B), R_{22}(A,D)
- Which one is better ?



BCNF and Dependencies

Unit	Company	Product

FD's: Unit → Company; Company, Product → Unit
So, there is a BCNF violation, and we decompose.

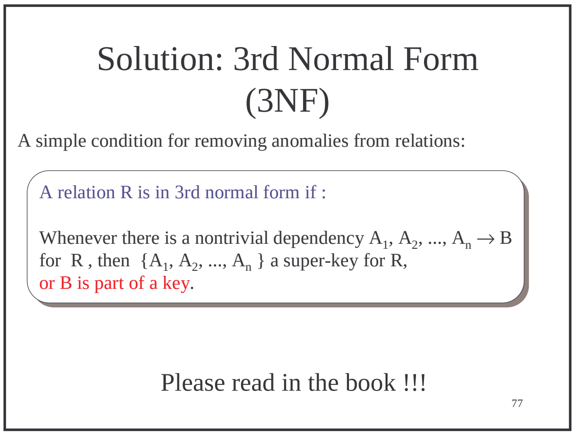
Unit	Company

Unit → Company

Unit	Product

No FDs

In BCNF we loose the FD: Company, Product à Unit



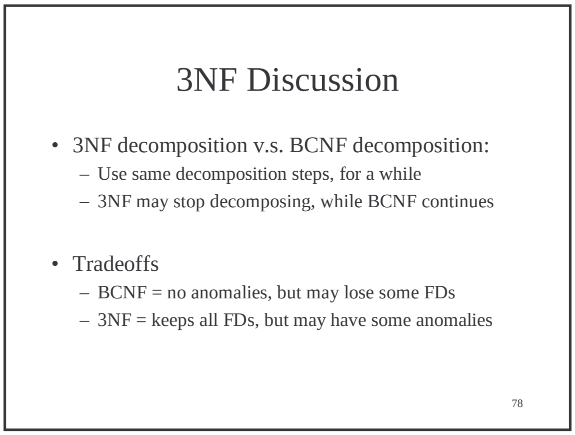
Solution: 3rd Normal Form (3NF)

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.

Please read in the book !!!



3NF Discussion

- 3NF decomposition v.s. BCNF decomposition:
  - Use same decomposition steps, for a while
  - 3NF may stop decomposing, while BCNF continues
- Tradeoffs
  - BCNF = no anomalies, but may lose some FDs
  - 3NF = keeps all FDs, but may have some anomalies