Lecture 2:
Database Modeling (end)
The Relational Data Model

April 11, 2002

1

Today's Attractions

- End of database modeling:
  - Subclasses (2.4)
  - Constraints: (2.5)
  - Weak entity sets (2.6)
- The relational data model
- From E/R diagrams to relations
- Start functional dependencies

2

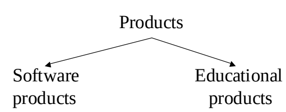
Administration

- Exam date: June 10th, anyone?
- Project groups

3

Modeling Subclasses

Some objects in a class may be special
- define a new class
- better: define a *subclass*



So --- we define subclasses (in ODL and in E/R).

4

Subclasses in ODL

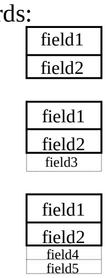
```
interface  SoftwareProduct:  Product{
      attribute  string   platform;
      attribute  integer  requiredMemory;
}

interface EducationalProduct: Product{
      attribute struct Interval {integer begin, integer end} ageGroup;
      attribute string topic;
}
```

The two classes *inherit* all the properties of Product.

5

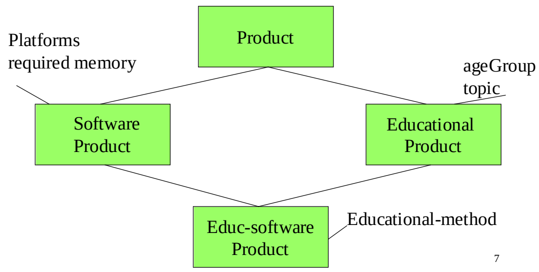
Understanding Subclasses

- Think in terms of records:
  - Product
  - SoftwareProduct
  - EducationalProduct



6

Multiple Inheritance in ODL



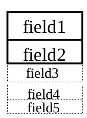
7

```
interface EducSoftwareProduct:
      SoftwareProduct, EducationalProduct {
          attribute string educational-method;
}
```

8

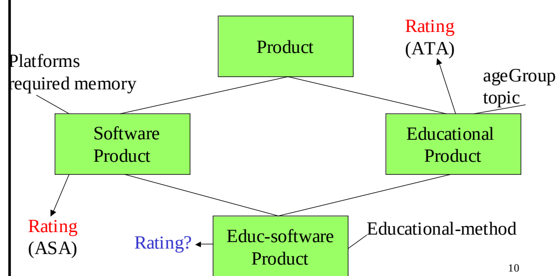
Understanding Multiple Inheritance

- Think in terms of records:
  - EducSoftwareProduct



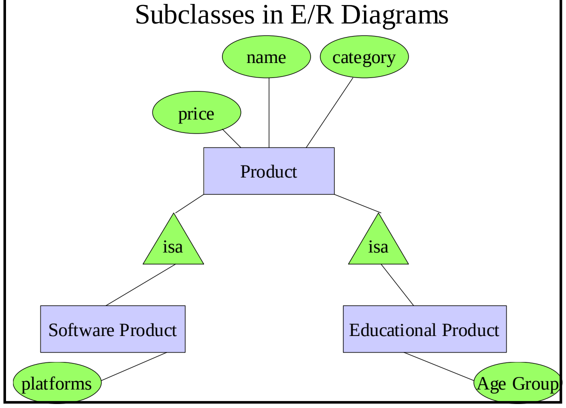
9

How do we resolve conflicts?



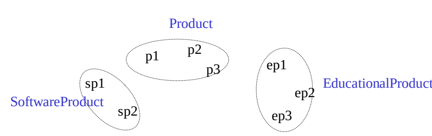
10

Subclasses in E/R Diagrams



Difference between ODL and E/R inheritance

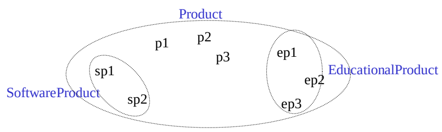
- ODL: classes are disjoint



12

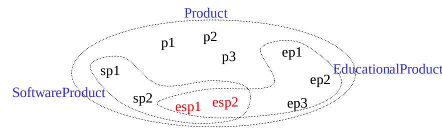
Difference between ODL and E/R inheritance

- E/R: entity sets overlap



13

- No need for multiple inheritance in E/R



- we have three entity sets, but four different kinds of objects

14

Constraints

- A constraint = an assertion about the database that must be true at all times
- part of the db schema
- types in programming languages do not have anything similar
- correspond to *invariants* in programming languages

15

Modeling Constraints

Finding constraints is part of the modeling process.
Commonly used constraints:

Keys: social security number uniquely identifies a person.

Single-value constraints: a person can have only one father.

Referential integrity constraints: if you work for a company, it must exist in the database.

Domain constraints: peoples' ages are between 0 and 150.
General constraints: all others (at most 50 students enroll in a class)

16

Keys

A set of attributes that uniquely identify an object or entity:

Person: social-security-number
name
name + address
name + address + age

Perfect keys are often hard to find, so organizations usually invent something.

17

Keys

- Multi-attribute keys:
  - E.g. name + address
- Multiple keys:
  - E.g social-security-number, name + address

18

Keys in ODL

```
interface Person
      (key  ssn)
  {
      attribute string ssn;
      attribute string name;
      ...
  }
```

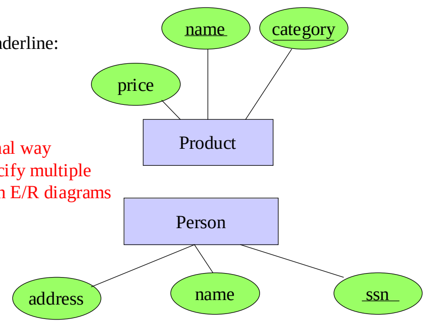
Defining multiple keys:

(**key** ssn employeID (name address age))

19

Keys in E/R Diagrams

Underline:



No formal way to specify multiple keys in E/R diagrams

20

Single Value Constraints

- Sometimes we can choose to allow one or more values
- ODL:
  - attributes are always single value
  - relationships have single or multiple values

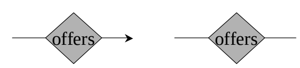
    relationship person president;

    relationship set<person> presidents;

21

Single Value Constraints

- E/R:



22

Single Value Constraints

Single Value Constraint:

- we explicitly require one value
- two flavors:
  - allow nulls
  - do not allow nulls

23

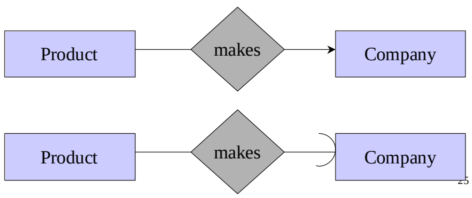
Referential Integrity Constraints

- In some formalisms we may refer to other object but get garbage instead
  - e.g. a dangling pointer in C/C++
- the Referential Integrity Constraint explicitly requires a reference to exist.

24

Referential Integrity Constraints

- In E/R:



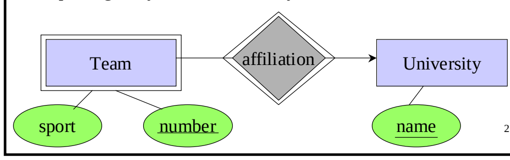
25

Weak Entity Sets

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

This happens if:

- part-of hierarchies
- splitting n-ary relations to binary.



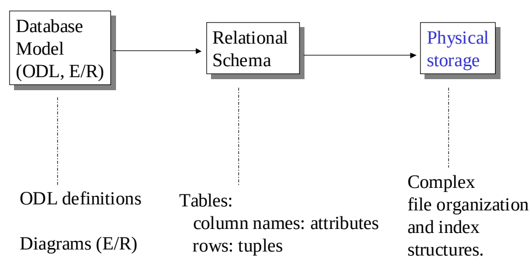
26

The Relational Model: Outline

- The relational model (3.1)
- E/R to relational model (3.2)
- Subclasses to relational model (3.3)
- *ODL to relational model (read on your own, section 4.4).*

27

The Relational Data Model



28

Terminology

Table name

Attribute names

Products:

Name	Price	Category	Manufacturer
gizmo	$19.99	gadgets	GizmoWorks
Power gizmo	$29.99	gadgets	GizmoWorks
SingleTouch	$149.99	photography	Canon
MultiTouch	$203.99	household	Hitachi

tuples

29

Domains

- each attribute has a type
- must be atomic type (why ? see later)
- called *domain*
- examples:
  - Integer
  - String
  - Real
  - …

30

Schemas

Relational Schema:
- Relation name plus attribute names
- E.g. Product(Name, Price, Category, Manufacturer)
- In practice we add the domain for each attribute

Database Schema
- Set of relational schemas
- E.g. Product(Name, Price, Category, Manufacturer),
  Vendor(Name, Address, Phone),
  . . . .

31

Instances

- **Relational schema** = R(A1,…,Ak):
  Instance = relation with k attributes (of "type" R)
  - values of corresponding domains
- **Database schema** = R1(…), R2(…), …, Rn(…)
  Instance = n relations, of types R1, R2, ..., Rn

32

Example

Relational schema: Product(Name, Price, Category, Manufacturer)
Instance:

Name	Price	Category	Manufacturer
gizmo	$19.99	gadgets	GizmoWorks
Power gizmo	$29.99	gadgets	GizmoWorks
SingleTouch	$149.99	photography	Canon
MultiTouch	$203.99	household	Hitachi

33

Updates

The database maintains a current database state.

Updates to the data:

1) add a tuple
2) delete a tuple
3) modify an attribute in a tuple

Updates to the data happen very frequently.

Updates to the schema: relatively rare. Rather painful. Why**?**

34

Schemas and Instances

- Analogy with programming languages:
  - Schema = type
  - Instance = value
- Important distinction:
  - Database Schema = stable over long periods of time
  - Database Instance = changes constantly, as data is inserted/updated/deleted

35

Two Mathematical Definitions of Relations

Relation as cartesian product
- Tuple = element of **string** x **int** x **string** x **string**
- E.g. t = (gizmo, 19, gadgets, GizmoWorks)
- Relation = subset of **string** x **int** x **string** x **string**
- Order in the tuple is important !
  - (gizmo, 19, gadgets, GizmoWorks)
  - (gizmo, 19 , GizmoWorks, gadgets)
- No attributes

36

Relation as a set of functions

- Fix the set of attributes
  - A={name , price, category, manufacturer}
- A tuple = function t:A → Domains
- Relation = set of tuples
- E.g.

{name → gizmo,
price → 19,
category → gadgets,
manufacturer → gizmoWorks}

- Order in a tuple is not important
- Attribute names are important

37

Two Definitions of Relations

- We will switch back and forth between these two:
  - Positional tuples, without attribute names
  - Relational schemas with attribute names

38

From E/R Diagrams to Relational Schema

Easier than ODL (using a liberal interpretation of the word "easy")

- relationships are already independent entities

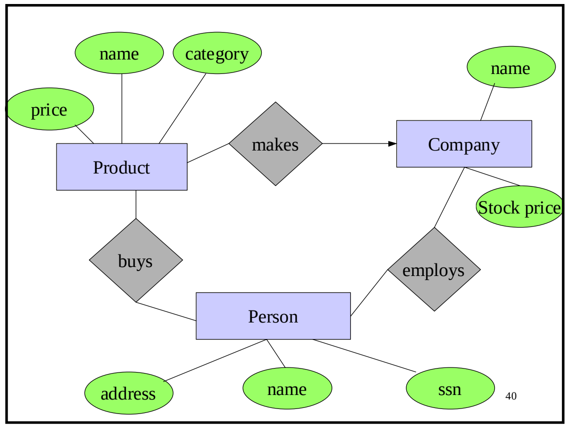
- only atomic types exist in the E/R model.

Entity sets → relations

Relationships → relations

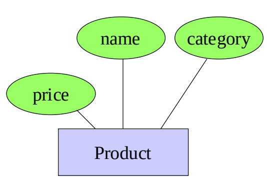
Special care for weak entity sets.

39



40

Entity Sets to Relations

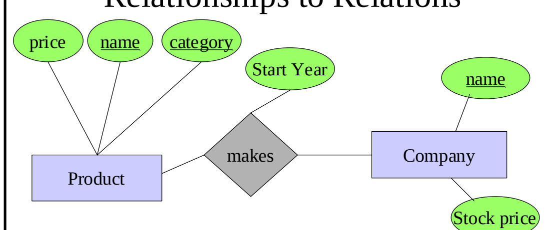


Product:

Name	Category	Price
gizmo	gadgets	$19.99

41

Relationships to Relations

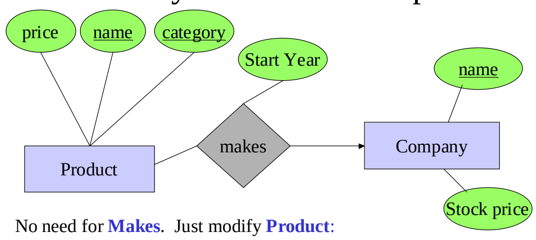


Relation **Makes** (watch out for attribute name conflicts)

Product-name	Product-Category	Company-name	Starting-year
gizmo	gadgets	gizmoWorks	1963

42

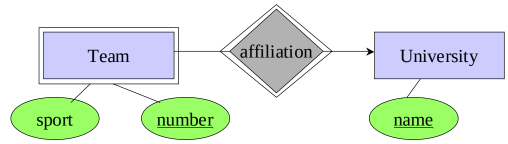
Many-one Relationships



No need for **Makes**. Just modify **Product**:

name	category	price	StartYear	companyName
gizmo	gadgets	19.99	1963	gizmoWorks

Handling Weak Entity Sets

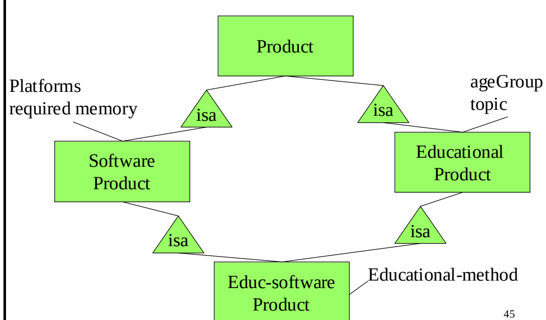


Relation Team:

Sport	Number	Affiliated University
mud wrestling	15	Montezuma State U.

- need all the attributes that contribute to the key of Team
- don't need a separate relation for Affiliation. (why ?)

Modeling Subclass Structure



Option #1: the "ODL" Approach

4 tables: each object can only belong to a single table

Product(<u>name</u>, price, category, manufacturer)

EducationalProduct(<u>name</u>, price, category, manufacturer, ageGroup, topic)

SoftwareProduct(<u>name</u>, price, category, manufacturer, platforms, requiredMemory)

EducationalSoftwareProduct(<u>name</u>, price, category, manufacturer, ageGroup, topic, platforms, requiredMemory)

All <u>names</u> are distinct

Option #2: the E/R Approach

Product(<u>name</u>, price, category, manufacturer)

EducationalProduct(<u>name</u>, ageGroup, topic)

SoftwareProduct(<u>name</u>, platforms, requiredMemory)

No need for a relation EducationalSoftwareProduct

Unless, it has a specialized attribute:
EducationalSoftwareProduct(<u>name</u>, educational-method)

Same <u>name</u> may appear in several relations

Option #3: The Null Value Approach

Have one table:

Product (name, price, manufacturer, age-group, topic, platforms, required-memory, educational-method)

Some values in the table will be NULL, meaning that the attribute not make sense for the specific product.

Too many meanings for NULL

Relational Schema Design

- Normalization: general idea
- Functional dependencies: the tool
- Using FD's for normalization.

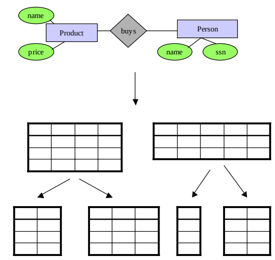
49

Relational Schema Design

Conceptual Model:

Relational Model:
(plus FD's)

Normalization:



50

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

51

Functional Dependencies

Definition:

If two tuples agree on the attributes

$A_1, A_2, \ldots A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots B_m$

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

Main (and simplest) example: keys

52

Examples

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

- EmpID $\longrightarrow$ Name, Phone, Position
- Position $\longrightarrow$ Phone
- but Phone $\not\longrightarrow$ Position

53

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

54

Functional Dependencies

Definition:

If two tuples agree on the attributes

$A_1, A_2, \ldots A_n$

then they must also agree on the attributes

$B_1, B_2, \ldots B_m$

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

Main (and simplest) example: keys

How many different FDs are there?

55

Examples

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

- EmpID $\rightarrow$ Name, Phone, Position
- Position $\rightarrow$ Phone
- but Phone $\not\rightarrow$ Position

56

In General

- To check A $\rightarrow$ B, erase all other columns

…	A	…	B	
	X1		Y1	
	X2		Y2	
	…		…	

- check if the remaining relation is many-one (called ***functional*** in mathematics)

57

Example

EmpID	Name	Phone	Position
E0045	Smith	1234 $\leftarrow$	Clerk
E1847	John	9876 $\leftarrow$	Salesrep
E1111	Smith	9876 $\leftarrow$	Salesrep
E9999	Mary	1234 $\leftarrow$	lawyer

58

More Examples

Product: name $\rightarrow$ price, manufacturer
Person: ssn $\rightarrow$ name, age
Company: name $\rightarrow$ stock price, president

Key of a relation is a set of attributes that:

- functionally determines all the attributes of the relation
- none of its subsets determines all the attributes.

Superkey: a set of attributes that contains a key.

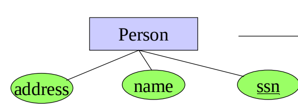
59

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.



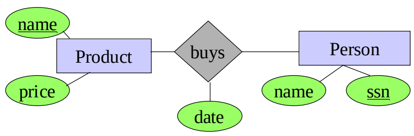
$\rightarrow$ **Person(**address, name, <u>ssn</u>**)**

60

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

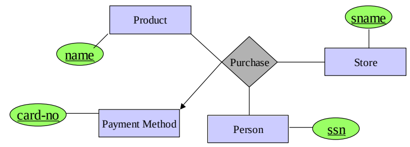


buys(<u>name, ssn</u>, date)

61

Finding the Keys

But: 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)

62

Finding the Keys

More rules:

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

(Try to find them yourself, check book)

63

Rules for FD's

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

Is equivalent to

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

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

…

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

Splitting rule and Combing rule

64

Rules in FD's (continued)

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

Trivial Rule

Why ?

65

Rules in FD's (continued)

Transitive Closure Rule

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

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

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

Why ?

66

Closure of a set of Attributes

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

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

67

Closure Algorithm

Start with X={A1, …, An}.

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

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

then

add C to X.

68

Example

A B $\rightarrow$ C
A D $\rightarrow$ E
B $\rightarrow$ D
A F $\rightarrow$ B

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

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

69

Why Is the Algorithm Correct ?

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

70

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

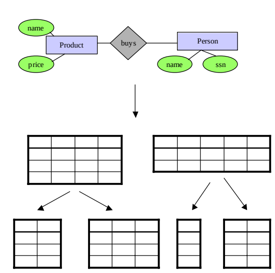
71

Relational Schema Design

Conceptual Model:

Relational Model:
(plus FD's)

Normalization:



72

Relational Schema Design

Recall set attributes (persons with several phones):

Name	SSN	Phone Number
Fred	123-321-99	(201) 555-1234
Fred	123-321-99	(206) 572-4312
Joe	909-438-44	(908) 464-0028
Joe	909-438-44	(212) 555-4000

Note: SSN no longer a key here

Goal: try to reduce anomalies:

Redundancy = repeat data

update anomalies = need to update in many places

deletion anomalies = need to delete many tuples

73

Relation Decomposition

Break the relation into two:

SSN	Name
123-321-99	Fred
909-438-44	Joe

SSN	Phone Number
123-321-99	(201) 555-1234
123-321-99	(206) 572-4312
909-438-44	(908) 464-0028
909-438-44	(212) 555-4000

74

Decompositions in General

Let *R* be a relation with attributes $A_1, A_2, \ldots A_n$

Create two relations *R1* and *R2* with attributes

$B_1, B_2, \ldots B_m$ $\quad$ $C_1, C_2, \ldots C_l$

Such that:

$B_1, B_2, \ldots B_m \cup C_1, C_2, \ldots C_l = A_1, A_2, \ldots A_n$

And

-- *R1* is the projection of *R* on $B_1, B_2, \ldots B_m$

-- *R2* is the projection of *R* on $C_1, C_2, \ldots C_l$

75

Incorrect Decomposition

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

Decompose on : **Name, Category** and **Price, Category**

Name	Category
Gizmo	Gadget
OneClick	Camera
DoubleClick	Camera

Price	Category
19.99	Gadget
24.99	Camera
29.99	Camera

When we put it back:

Cannot recover information

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

76

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...

77

Boyce-Codd Normal Form

A simple condition for removing anomalies from relations:

A relation R is in BCNF if and only if:

Whenever there is a nontrivial dependency $A_1, A_2, \ldots A_n \rightarrow B$ for R , it is the case that $\{ A_1, A_2, \ldots A_n \}$ a super-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*.

78

Example

Name	SSN	Phone Number
Fred	123-321-99	(201) 555-1234
Fred	123-321-99	(206) 572-4312
Joe	909-438-44	(908) 464-0028
Joe	909-438-44	(212) 555-4000

What are the dependencies?

SSN $\rightarrow$ Name

What are the keys?

Is it in BCNF?

79

Decompose it into BCNF

SSN	Name
123-321-99	Fred
909-438-44	Joe

SSN $\rightarrow$ Name

SSN	Phone Number
123-321-99	(201) 555-1234
123-321-99	(206) 572-4312
909-438-44	(908) 464-0028
909-438-44	(212) 555-4000

80

What About This?

Name	Price	Category
Gizmo	$19.99	gadgets
OneClick	$24.99	camera

Name $\rightarrow$ Price, Category

81

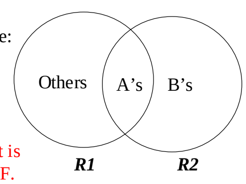
BCNF Decomposition

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:



Find a 2-attribute relation that is not in BCNF.

Continue until there are no BCNF violations left.

82

Example Decomposition

Person:

Name	SSN	Age	EyeColor	PhoneNumber

Functional dependencies:

SSN $\longrightarrow$ Name, Age, Eye Color

BNCF: Person1(SSN, Name, Age, EyeColor),
Person2(SSN, PhoneNumber)

What if we also had an attribute Draft-worthy, and the FD:

Age $\longrightarrow$ Draft-worthy

83