Lecture 2: E/R Diagrams and the Relational Model

Thursday, January 4, 2001

Outline

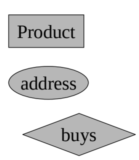
- E/R Diagrams: 2.4 (except 2.4.2, 2.4.5), 2.5 (except 2.5.4)
- The Relational Model: 3.1, 3.2, 3.3
- E/R to Relational: 3.5

Entity / Relationship Diagrams in Summary

Entity sets:

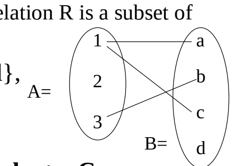
Properties:

Relationships:

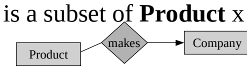


What is a Relation ?

- A mathematical definition:
  – if A, B are sets, then a relation R is a subset of A x B
- A={1,2,3}, B={a,b,c,d}, R = {(1,a), (1,c), (3,b)}

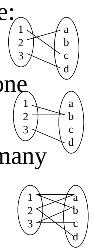


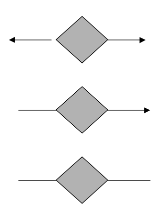
- **makes** is a subset of **Product** x **Company**:



Multiplicity of E/R Relations

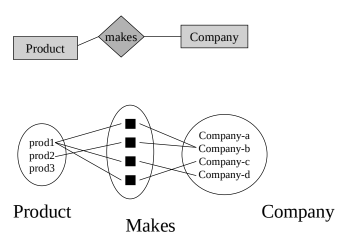
- one-one:
- many-one
- many-many





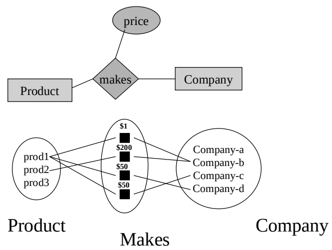
Note: the book uses a non-standard notation

E/R Relationships and Relations in Mathematics



Makes = a subset of Product x Company

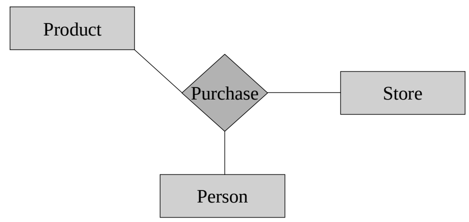
Attributes on Relationshiops



Makes = still a subset of Product x Company,
not of Product x Company x Price (what does this imply ?)

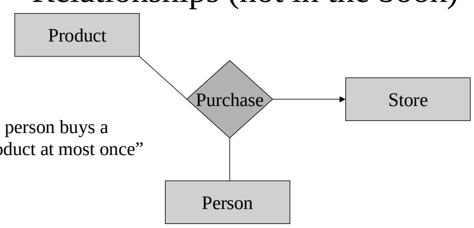
Multi-way Relationships

How do we model a purchase relationship between buyers, products and stores?



Can still model as a mathematical set (how ?)

Arrows in Multi-way Relationships (not in the book)

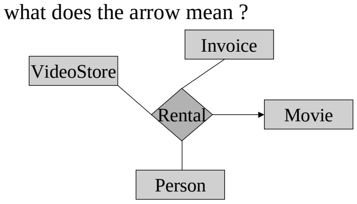


“A person buys a product at most once”

Limited expressive power.
Cannot say: “a person buys at most one product”

Arrows in Multiway Relationships

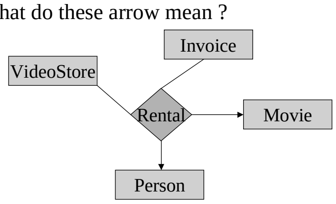
- Q: what does the arrow mean ?



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

Arrows in Multiway Relationships

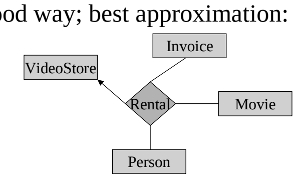
- Q: what do these arrow mean ?



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

Arrows in Multiway Relationships

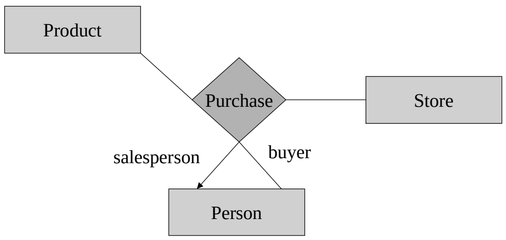
- Q: how do I say: “invoice determines store” ?
- A: no good way; best approximation:



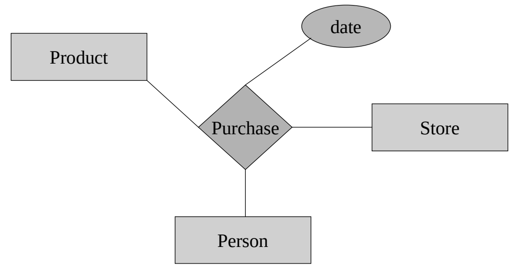
- Why is this incomplete ?

Roles in Relationships

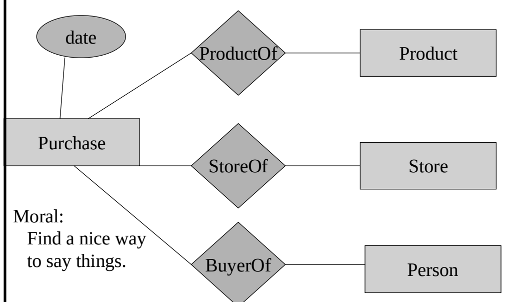
What if we need an entity set twice in one relationship?



Converting Multi-way Relationships to Binary



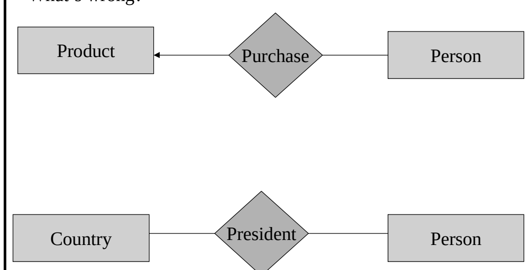
Converting Multi-way Relationships to Binary



Moral:
Find a nice way
to say things.

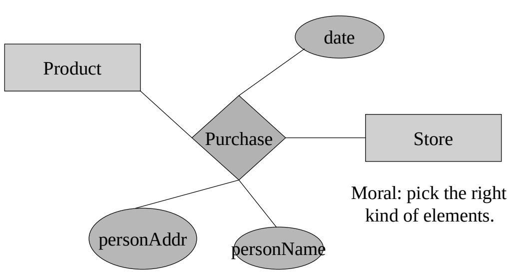
Design Principles

What's wrong?



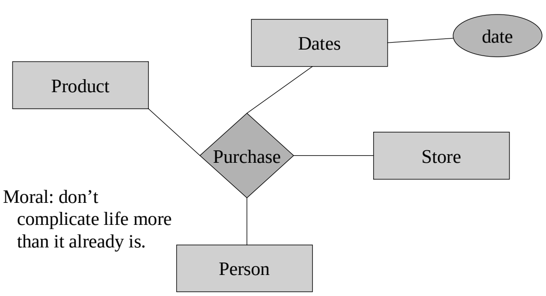
Moral: be faithful!

What's Wrong?



Moral: pick the right
kind of elements.

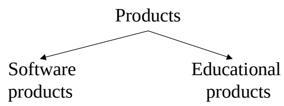
What's Wrong?



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

Class Hierarchies

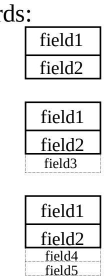
Some objects (entities) in a class may be special
- define a new class
- better: define a *subclass*



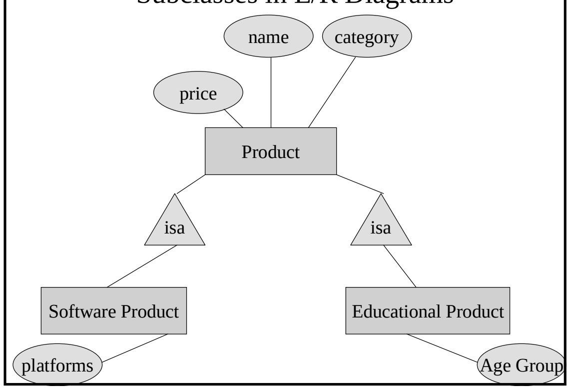
We define subclasses in E/R

Understanding Subclasses

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

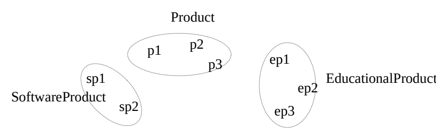


Subclasses in E/R Diagrams



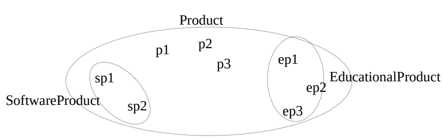
Difference between C++ and E/R inheritance

- C++: classes are disjoint

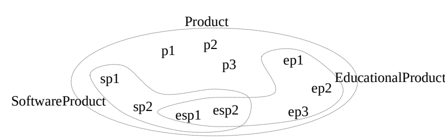


Difference between ODL and E/R inheritance

- E/R: entity sets overlap

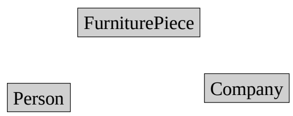


- No need for multiple inheritance in E/R



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

Modeling Union Types With Subclasses

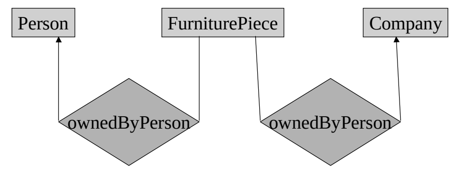


Say: each piece of furniture is owned either by a person, or by a company

Modeling Union Types with Subclasses

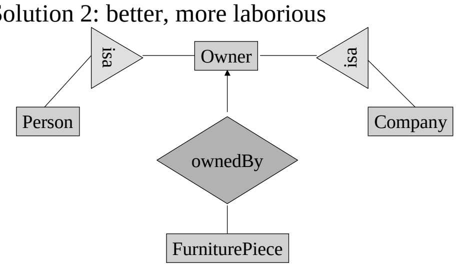
Say: each piece of furniture is owned either by a person, or by a company

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



Modeling Union Types with Subclasses

Solution 2: better, more laborious



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

Common Constraints in E/R

Keys: social security number uniquely identifies a person.

Single-value constraints: e.g. one-one, one-many, many-one

Participation constrain: total many-one

Domain constraints: peoples’ ages are between 0 and 150.

General constraints: all others (at most 50 students enroll in a class)

Keys

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

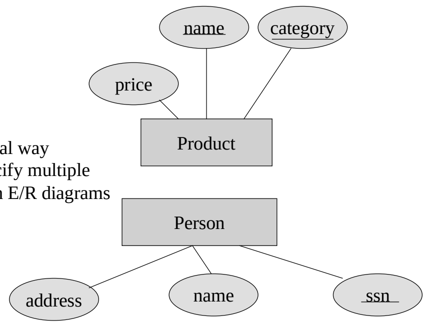
Person: ssn
name
name + address
name + address + age

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

An object may have multiple keys:

employee number, ssn

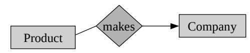
Keys in E/R Diagrams



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

Single Value Constraints in E/R

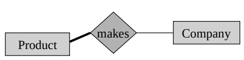
A product is made by at most one company:



Notice: some products are not made by any company

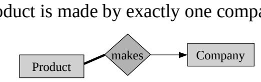
Participation Constraint

Each product is made by a lest one company



(notation from the book)

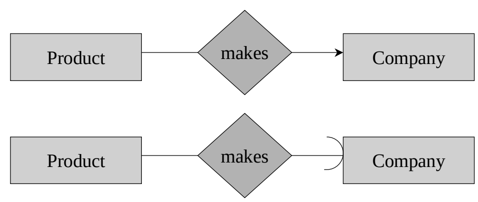
Each product is made by exactly one company



This: also called *referential integrity constraint*

Referential Integrity Constraint

- Another notation (in Ullman's book):

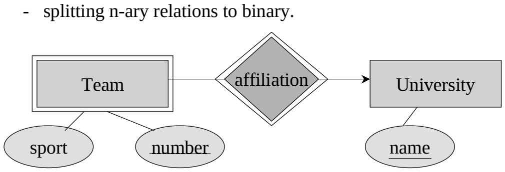


Weak Entity Sets

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.



The Relational Data Model

Database Model (E/R) → Relational Schema → Physical storage

Diagrams (E/R)

Tables:
column names: attributes
rows: tuples

Complex
file organization
and index
structures.

Terminology

Table name

Products:

Attribute names

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

Domains

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

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)

Instances

- An instance of a relational schema R(A1,…,Ak), is a relation with k attributes with values of corresponding domains
- An instance of a database schema R1(…), R2(…), …, Rn(…), consists of n relations, each an instance of the corresponding relational schema.

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

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

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)
  - Product is 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

Two Mathematical Definitions of Relations

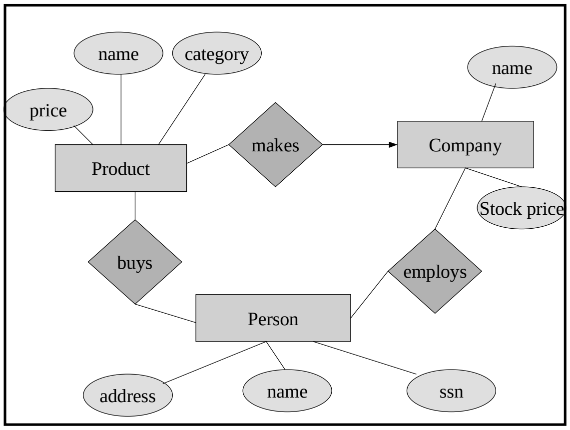
- Relation as a set of functions
  - Fix the set of attributes
    - A={name , price, category, manufacturer}
  - A tuple is a function t:A → Domains
  - Relation = set of tuples
  - E.g.
    {name → gizmo,
    price → 19,
    category → gadgets,
    manufacturer → gizmoWorks}
  - Order in a tuple is not important
  - Attribute names are important

Two Definitions of Relations

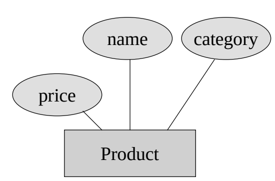
- We will switch back and forth between these two:
  - Relational schemas with attribute names
  - Positional tuples, without attribute names

From E/R Diagrams to Relational Schema

- Each entity set → relation
- Each relationship → relation
- Special cases:
  - one-one, many-one relationships
  - subclasses
  - weak entity sets



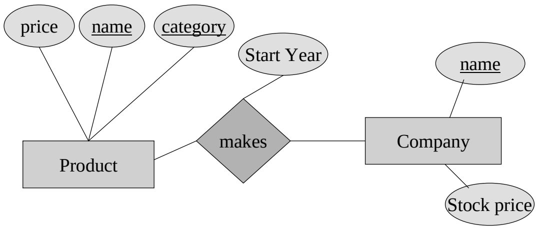
Entity Sets to Relations



Product:

Name	Category	Price
gizmo	gadgets	$19.99

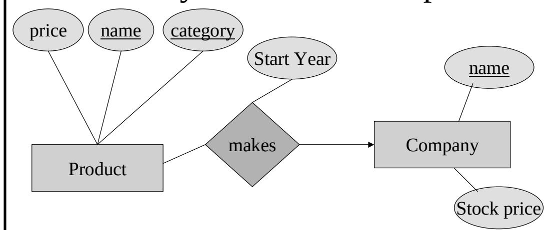
Relationships to Relations



Relations: Product, Makes, Company:

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

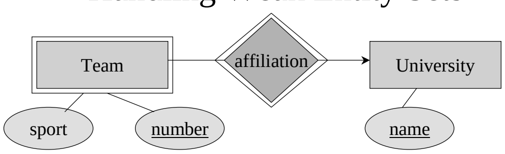
Many-one Relationships



No need for Makes. Just modify Product:

name	category	price	StartYear	companyName
gizmo	gadgets	19.99	1963	gizmoWorks

Handling Weak Entity Sets

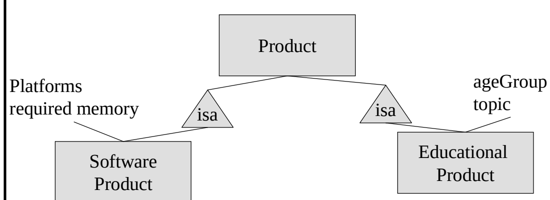


Relation Team:

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



The right way

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

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

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

Notice: each subclass stores only the *additional* attributes

Option #2: 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.