CS411
Database Systems

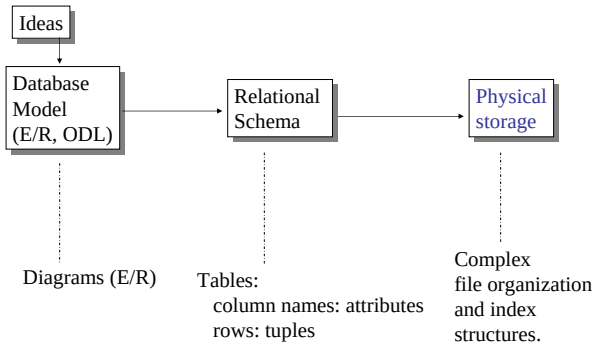
03: The Relational Model

Kazuhiro Minami

Announcements

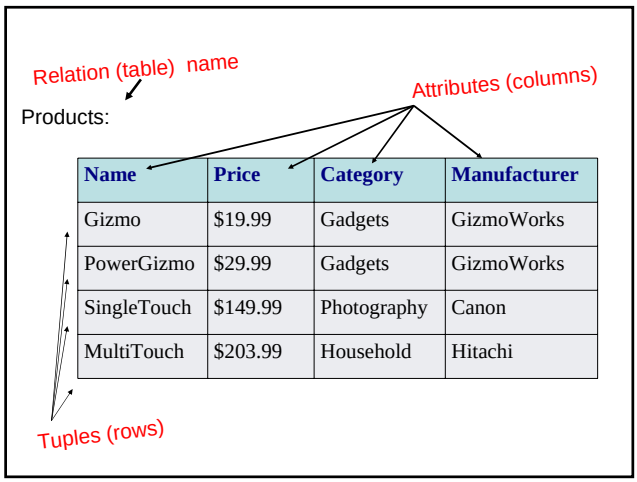
- Project stage 0 is due today
- Grade distribution of the course project
  - Stage 1 (Decide your application): 5%
  - Stage 2 (ER modeling) : 5%
  - Stage 3 (Relational schema design): 10%
  - Stage 4 (Demo for basic functions) : 30%
  - Stage 5 (Final demo/report): 50%
- Grade distribution of the graduate project
  - Stage A (Decide your survey topic): 5%
  - Stage B (Preliminary report): 30%
  - Stage C (Final report): 65%

DB Modeling & Implementation



Why do we need both the ER and relational models?

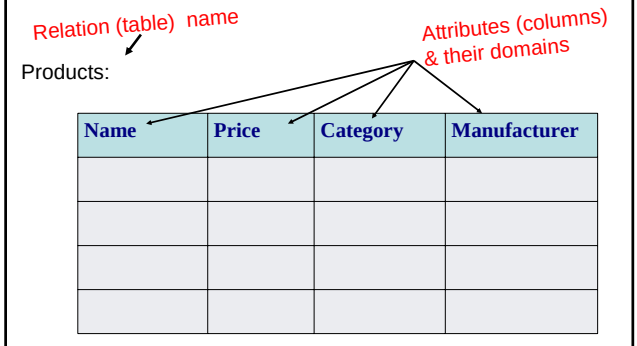
- Relational model has just a single concept: ***tables***
  - Allow us to express queries at a very high level
  - well-suited for efficient manipulations on computers
- ER model is richer: entities, relationships, attributes, etc.
  - well-suited for capturing application requirements
  - not so well-suited for computer implementation (no query language)



Each attribute has a type: its *domain*

- Integer, string, date, real
- Traditionally domains were not user-definable, e.g., “map”
- Domains must be atomic (why? see later)

The relation’s ***schema*** is its metadata



The relation’s ***instance*** = the data in it

Gizmo	$19.99	Gadgets	GizmoWorks
PowerGizmo	$29.99	Gadgets	GizmoWorks
SingleTouch	$149.99	Photography	Canon
MultiTouch	$203.99	Household	Hitachi

4 tuples (rows)

We can write a schema concisely:

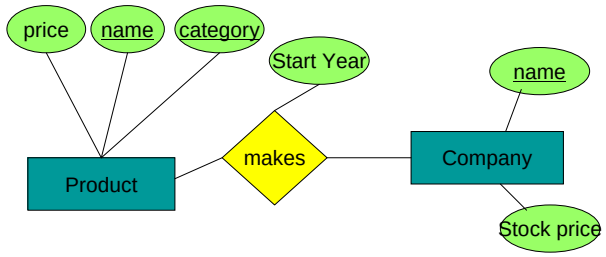
Product(Name, Price, Category, Manufacturer)

DB schema = finite set of relation schemas.

Product(Name, Price, Category, Manufacturer),
Vendor(Name, Address, Phone),
… .

Now the fun part: translating an ER diagram into the relational model

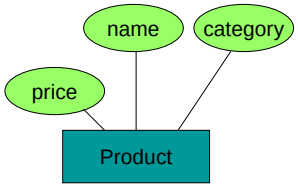
How to convert a E/R diagram to a set of relations?



How to translate an ER diagram to a relational schema

- Basic cases
  - Entity set E = relation with attributes of E
  - Relationship R = relation with attributes being keys of related entity sets + attributes of R
- Special cases
  - Combining two relations
  - Translating weak entity sets
  - Translating is-a relationships and subclasses

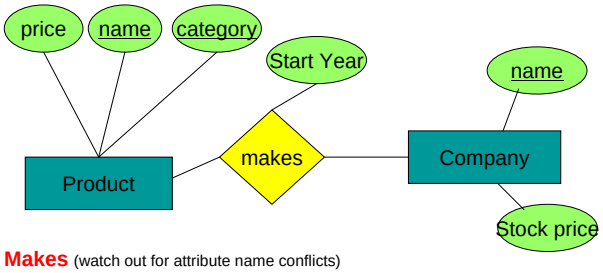
How to convert an entity set to a relation



Product:

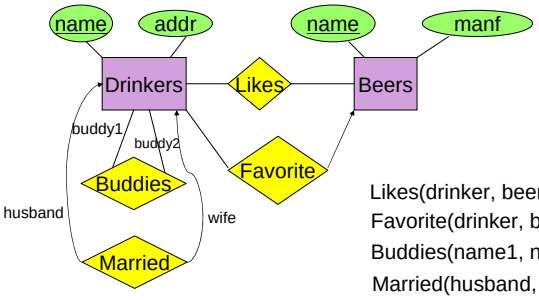
Name	Price	Category
Gizmo	$19.99	Gadgets

How to convert a relationship to a relation



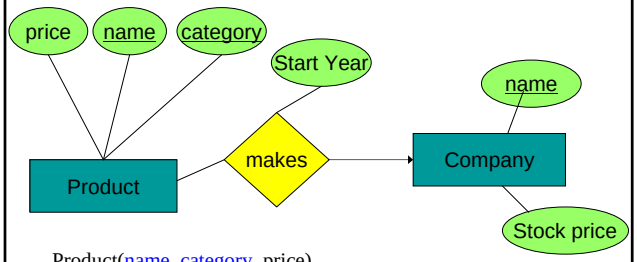
Makes (watch out for attribute name conflicts)

Product_ Name	Product_ Category	Company_ Name	Start_Year
Gizmo	Gadgets	GizmoWorks	1963



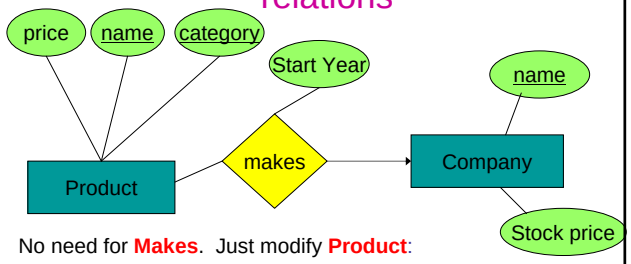
Likes(drinker, beer)
Favorite(drinker, beer)
Buddies(name1, name2)
Married(husband, wife)

Sometimes it is best to combine two relations



Product(name, category, price)
Makes(product_name, category, StartYear, company_name)
Company(name, StockPrice)

Sometimes it is best to combine two relations



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

Name	Price	Category	Manufacturer	Start_ Year
Gizmo	$19.99	Gadgets	GizmoWorks	1963

It is OK to combine the relation for an entity set *E* with the relation for a one-one relationship from *E* to another entity set.

Drinkers(name, addr)
Favorite(drinker, beer)

Drinkers(name, addr, favoriteBeer)

What if each drinker could have several favorite beers?

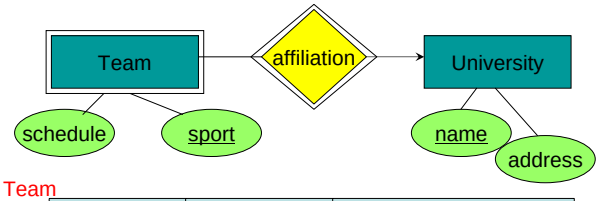
Combining a many-to-many relationship causes trouble

Companies(name, addr)
MakerOf(companyName, productType)

Name	Address	ProductType
GizmoWorks	123 MG Rd	Wheels
GizmoWorks	123 MG Rd	Whistles

Redundancy!

How to convert a weak entity set to a relation



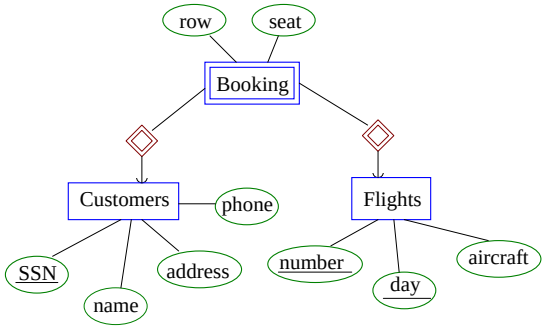
Team

Sport	Schedule	University_Name
Mud wrestling	<long string>	Montezuma State Univ.

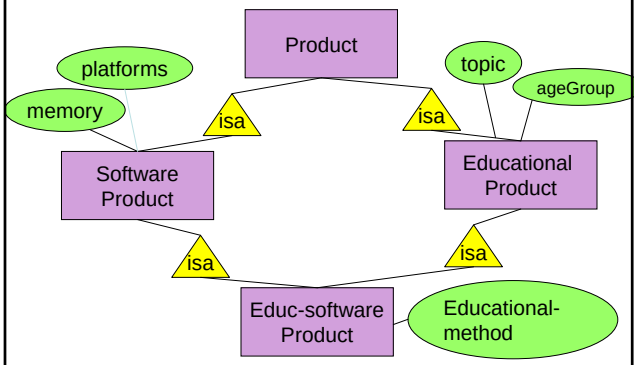
- **Do** include all the attributes in the key for Team.
- **Don't** include a separate relation for Affiliation. (why?)

Exercise 4.5.1

Convert the E/R diagram to a relational database schema.



How to translate a subclass



Three ways to translate subclasses

- *Object-oriented*: each entity belongs to exactly one class; create a relation for each class, with all its attributes.
- *E/R style*: create one relation for each subclass, with only the key attribute(s) and attributes attached to that entity set;
- *Use null*: create one relation; entities have nulls in attributes that don't belong to them.

Option #1: the OO Approach

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

Product(name, price, category, manufacturer)

EducationalProduct(name, price, category, manufacturer, ageGroup, topic)

SoftwareProduct(name, price, category, manufacturer, platforms, requiredMemory)

EducationalSoftwareProduct(name, price, category, manufacturer, ageGroup, topic, platforms, requiredMemory)

All names are distinct

Option #2: the E/R Approach

Product(name, price, category, manufacturer)

EducationalProduct(name, ageGroup, topic)

SoftwareProduct(name, platforms, requiredMemory)

No need for a relation EducationalSoftwareProduct unless it has a specialized attribute:
EducationalSoftwareProduct(name, educational-method)

Same name may appear in several relations

Option #3: The Null Value Approach

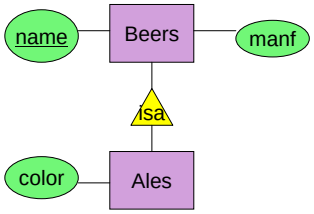
Have one table:

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

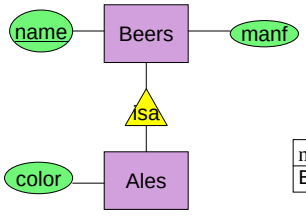
Some values in the table will be NULL, meaning that the attribute not make sense for the specific product.

Too many meanings for NULL

Example



Object Oriented

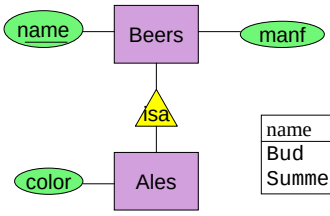


name	manf
Bud	Anheuser-Busch

Beers

name	manf	color
Summerbrew	Pete's	dark

Ales



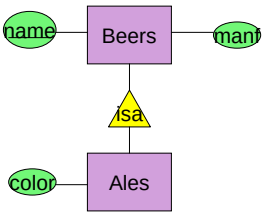
E/R Style

name	manf
Bud	Anheuser-Busch
Summerbrew	Pete's

Beers

name	color
Summerbrew	dark

Ales



Using Nulls

name	manf	color
Bud	Anheuser-Busch	NULL
Summerbrew	Pete's	dark

Beers

Comparisons

- O-O approach good for queries like "find the color of ales made by Pete's."
  - Just look in Ales relation.
- E/R approach good for queries like "find all beers (including ales) made by Pete's."
  - Just look in Beers relation.
- Using nulls saves space unless there are *lots* of attributes that are usually null.