Review: Logical Database Design

Alvin Cheung

Spring 2023

Many Steps in Database Design!

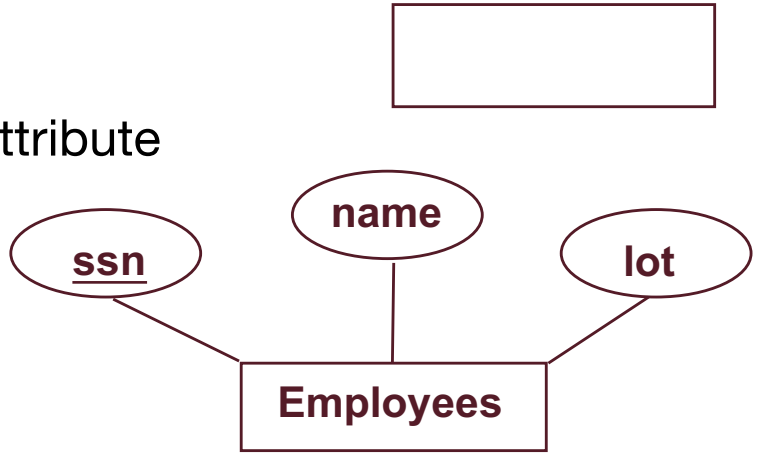
- **Requirements Analysis**
  - Translating user needs; what must database do? what must it capture?
- **Conceptual Design [Needs => ER Diagram]**
  - *high level visual description of data(often done w/ER model)*
  - Object-Relational Mappings (ORMs: Hibernate, Rails, Django, etc) encourage you to program here [essentially ER models]
- **Logical Design [ER Diagram => Relations]**
  - translate ER into DBMS data model
  - ORMs often require you to help here too
  - can be partially automated
- **Schema Refinement [Relations => Better Relations]**
  - consistency, normalization [add constraints, break or merge relations]
- **Physical Design [Storing Relations]**
  - indexes, disk layout

- **Orthogonal: Security Design [Relational Access Control]**
  - who accesses what, and how


We are here

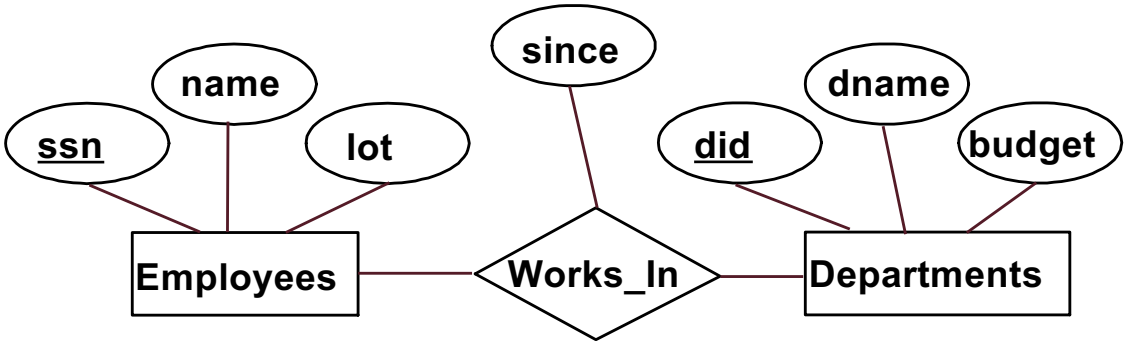
ER Model Basics: Entities

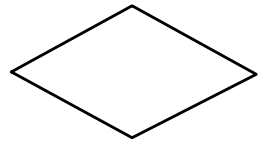
- **Entity**:
  - A real-world object described by a set of attribute values.



- **Entity Set:** A collection of similar entities.
  - E.g., all employees.
  - All entities in an entity set have the same attributes.
  - Each entity set has a primary key (underlined)
  - Each attribute has a domain

ER Model Basics: Relationships





Relationship: Association among two or more entities.

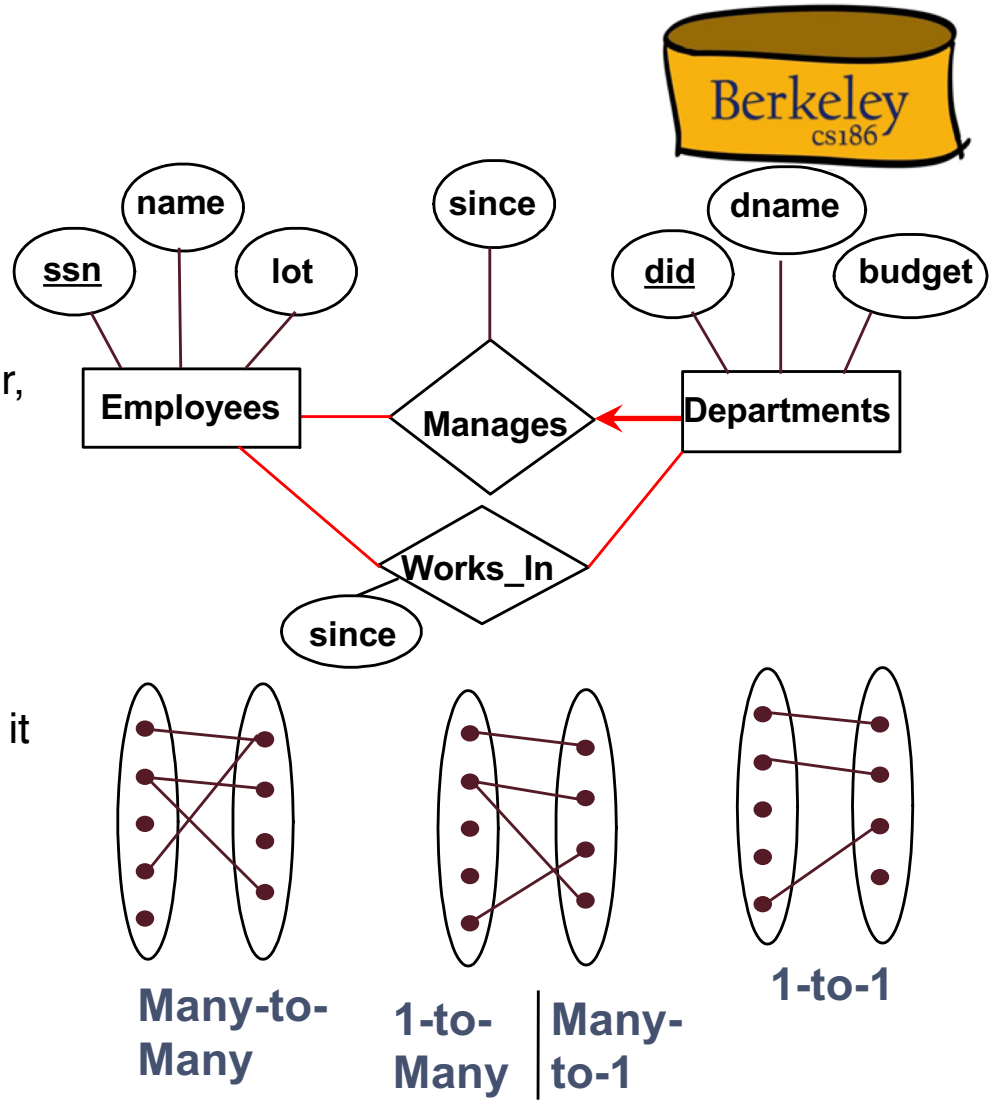
- E.g., Jenny (employee) works in Pharmacy department.
- Relationships can have their own attributes.

Relationship Set: Collection of similar relationships.

- An n-ary relationship set R relates n entity sets E_1 ... E_n; each relationship in R involves entities $e_1 \in E_1$, ..., $e_n \in E_n$

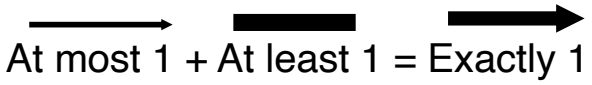
Key Constraints

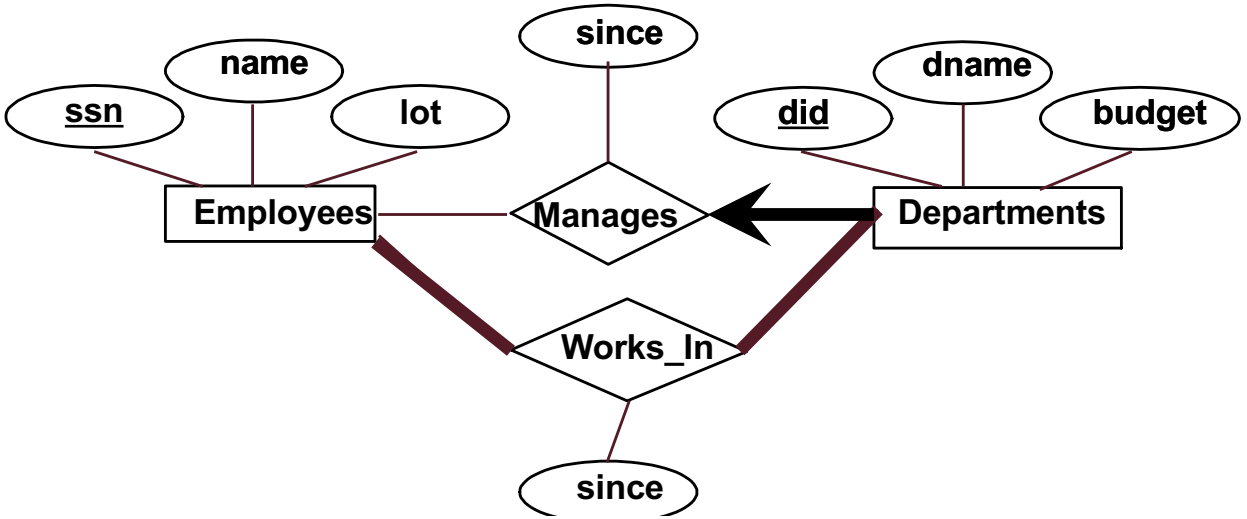
- An employee can work in **many** departments; a dept can have **many** employees.
- In contrast, each dept has **at most one** manager, according to the *key constraint* on **Dept** in the **Manages** relationship set. Equivalently:
  - Each dept participates at most once in this relationship
  - Each dept has at most one emp. managing it
- A **key constraint** gives a 1-to-many/many-to-1 relationship.



Participation Constraints

- Does every employee work in a department?
  - If so: a **participation constraint**
  - participation of Employees in Works_In is **total** (vs. partial)
  - Basically means that every employee participates in "**at least one**".
- Likewise, what if every department has an employee working in it?
- Likewise, what if every department has a manager?
  - Along with the arrow (at most one), this means exactly one

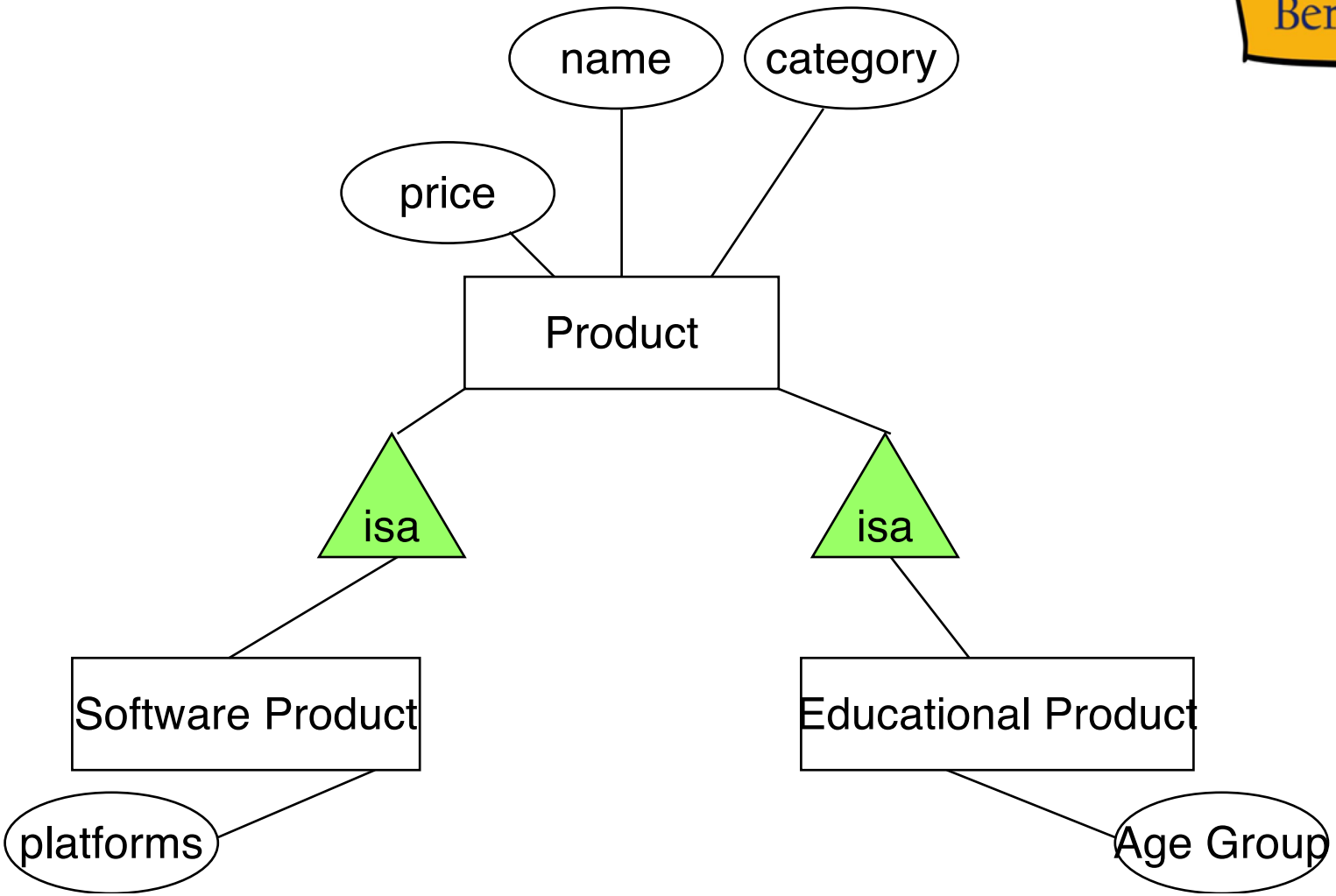




Recap: ER Diagrams – Constraints

- Key constraint 
  - at most one
- Participation constraint 
  - at least one
- Key constraint with total participation 
  - exactly one
- Non-key partial participation
  - 0 or more (no restrictions)

Subclasses in ER Diagrams



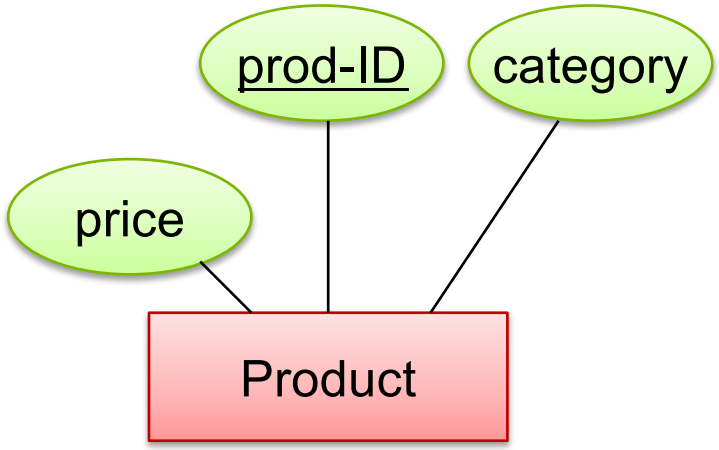
Steps in Database Design, Part 4

- Requirements Analysis
  - user needs; what must database do?
- Conceptual Design
  - *high level description (often done w/ER model)*
  - ORM encourages you to program here
- **Logical Design**
  - **translate ER into DBMS data model**
  - **ORMs often require you to help here too**
- Schema Refinement
  - consistency, normalization
- Physical Design - indexes, disk layout
- Security Design - who accesses what, and how

Completed

We are here

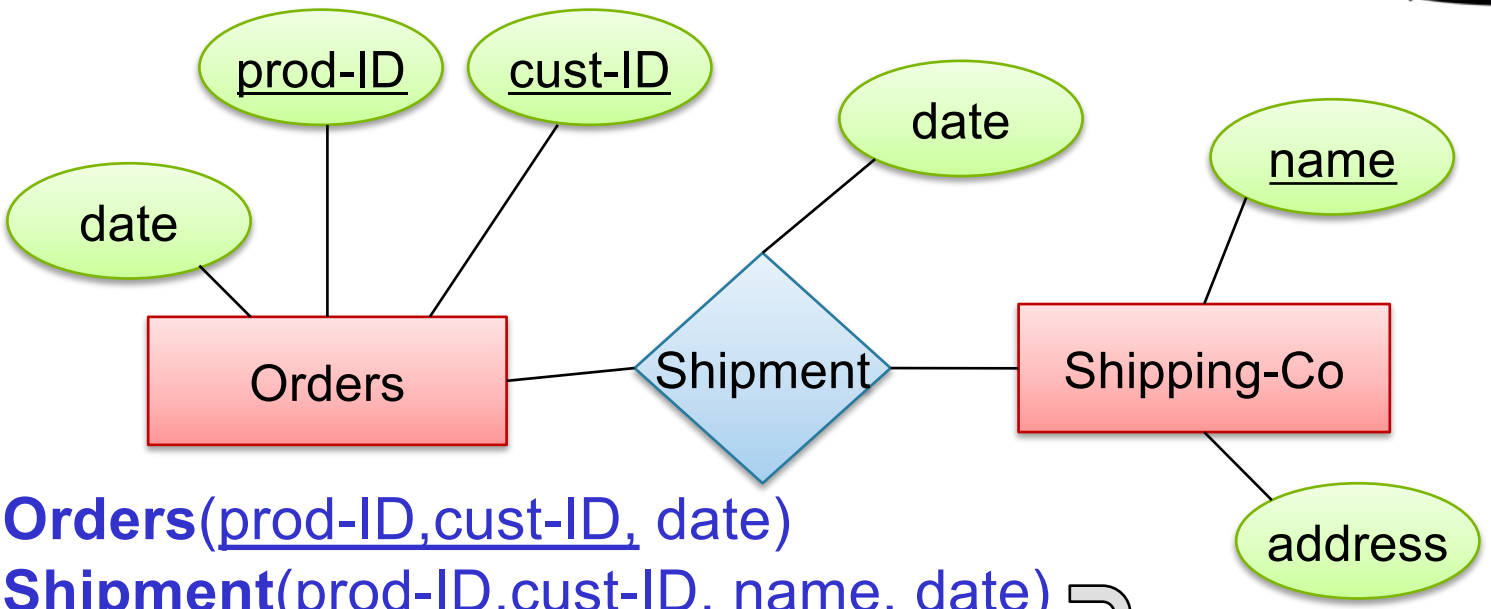
Entity Set to Relation



Product(prod-ID, category, price)

prod-ID	category	price
Gizmo55	Camera	99.99
Pokemn19	Toy	29.99

N-N Relationships to Relations

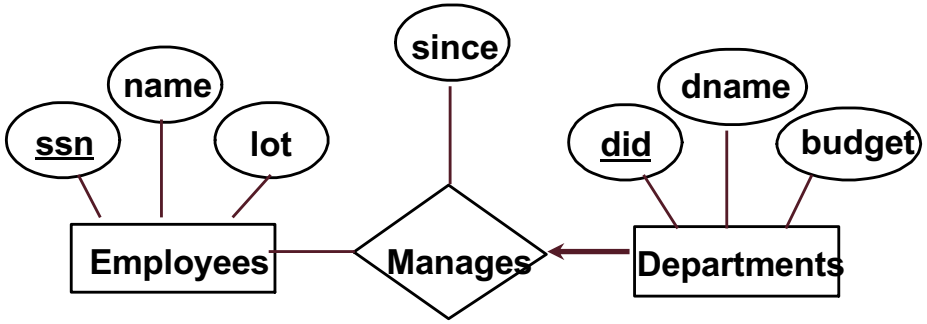


Orders(prod-ID,cust-ID, date)
Shipment(prod-ID,cust-ID, name, date)
Shipping-Co(name, address)

prod-ID	cust-ID	name	date
Gizmo55	Joe12	UPS	4/10/2011
Gizmo55	Joe12	FEDEX	4/9/2011

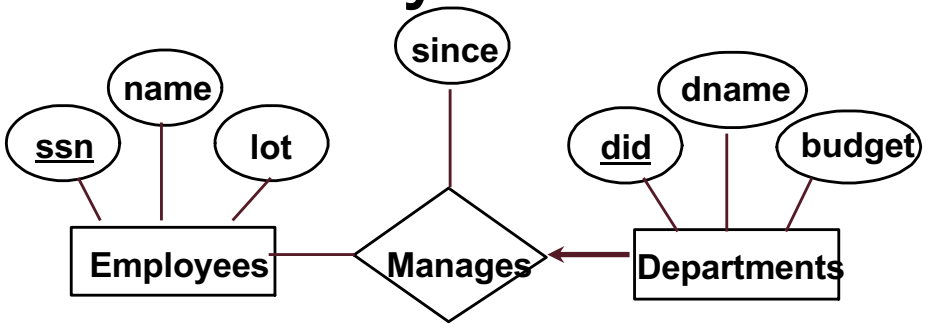
Note that keys from each participating entity set appear as foreign keys. This set of attributes forms a ***superkey*** for the relation.

Translating ER with Key Constraints




```
CREATE TABLE  Manages(
 ssn  CHAR(11),
 did  INTEGER,
 since  DATE,
 PRIMARY KEY (did),
 FOREIGN KEY (ssn)
   REFERENCES Employees,
 FOREIGN KEY (did)
   REFERENCES Departments)
```

Translating ER with Key Constraints



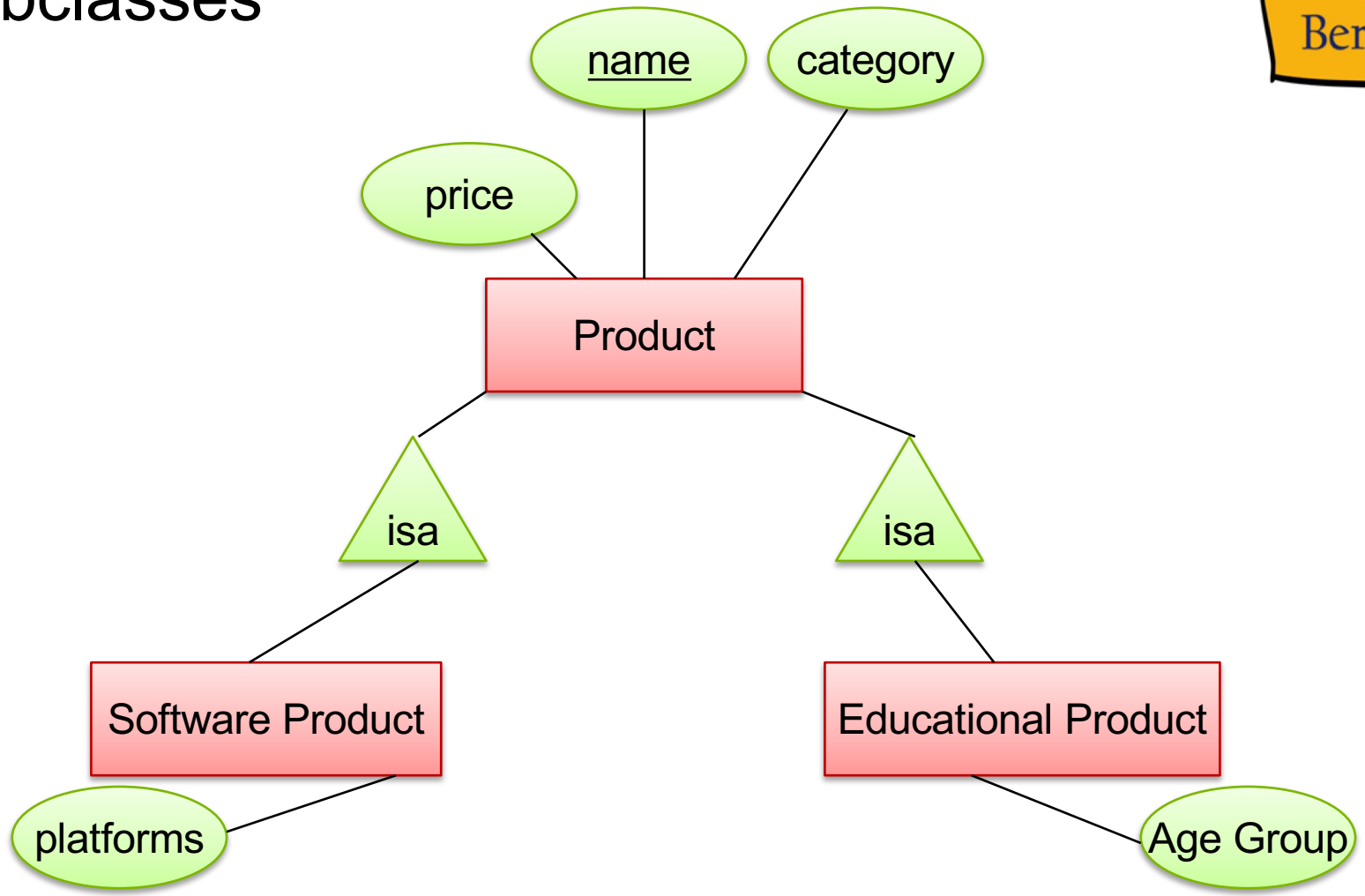
Since each department has a unique manager, we could instead combine Manages and Departments.

```
CREATE TABLE  Manages(
 ssn  CHAR(11),
 did  INTEGER,
 since  DATE,
 PRIMARY KEY (did),
 FOREIGN KEY (ssn)
   REFERENCES Employees,
 FOREIGN KEY (did)
   REFERENCES Departments)
```

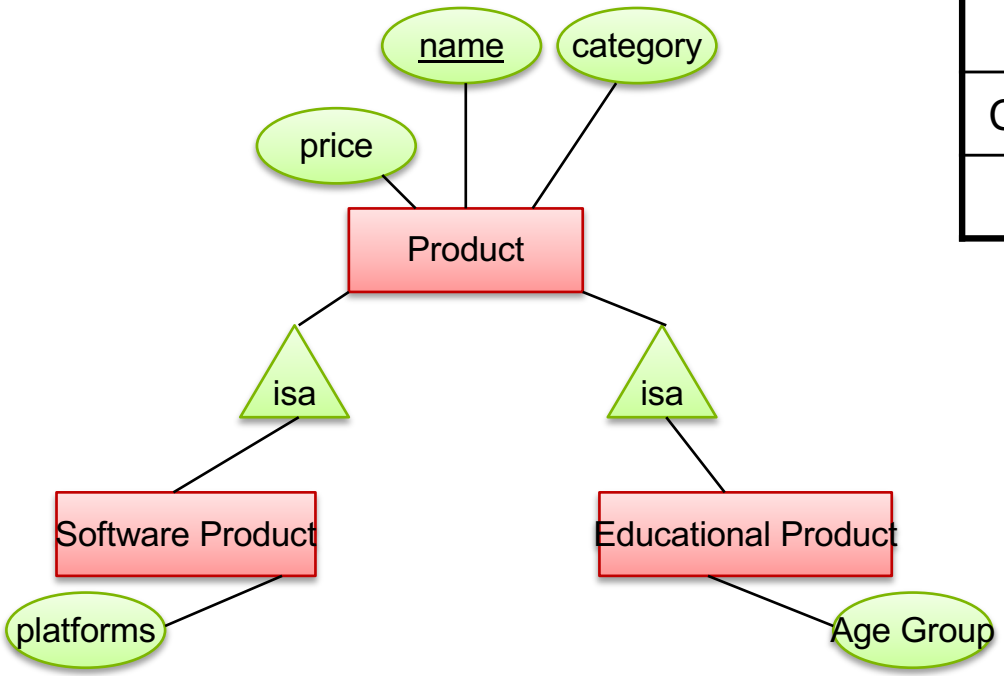
Vs.

```
CREATE TABLE  Dept_Mgr(
 did  INTEGER,
 dname  CHAR(20),
 budget  REAL,
 ssn  CHAR(11),
 since  DATE,
 PRIMARY KEY (did),
 FOREIGN KEY (ssn)
   REFERENCES Employees)
```

Subclasses


name
category
price
Product
isa
isa
Software Product
Educational Product
platforms
Age Group

Subclasses to Relations



Other ways to convert are possible

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	toddler
Toy	retired

Steps in Database Design, cont

- Requirements Analysis
  - user needs; what must database do?
- Conceptual Design
  - *high level description (often done w/ER model)*
  - ORM encourages you to program here

  Completed
- Logical Design
  - translate ER into DBMS data model
  - ORMs often require you to help here too

  Completed
- **Schema Refinement**
  - **consistency, normalization**

  You are here
- Physical Design - indexes, disk layout
- Security Design - who accesses what, and how

Relational Schema Design

Name	SSN	PhoneNumber	City
Fred	123-45-6789	510-555-1234	Berkeley
Fred	123-45-6789	510-555-6543	Berkeley
Joe	987-65-4321	908-555-2121	San Jose

Anomalies:

- Redundancy = repeat data
- Update anomalies = what if Fred moves to “Oakland”?
- Deletion anomalies = what if Joe deletes his phone number?

Relation Decomposition

Break the relation into two:

Name	SSN	PhoneNumber	City
Fred	123-45-6789	510-555-1234	Berkeley
Fred	123-45-6789	510-555-6543	Berkeley
Joe	987-65-4321	908-555-2121	San Jose

Name	SSN	City
Fred	123-45-6789	Berkeley
Joe	987-65-4321	San Jose

SSN	PhoneNumber
123-45-6789	510-555-1234
123-45-6789	510-555-6543
987-65-4321	908-555-2121

Anomalies have gone:

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

Relational Schema Design (or Logical Design)

How do we do this systematically?

- Start with some relational schema
- Find out its ***functional dependencies*** (FDs)
- Use FDs to ***normalize*** the relational schema

Functional Dependencies (FDs)

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:

$A_1 \ldots A_n$ **determines** $B_1..B_m$

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

Example

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

Finding New FDs: Armstrong's Axioms

- Suppose X, Y, Z are sets of attributes, then:
  - *Reflexivity*: If $X \supseteq Y$, then $X \rightarrow Y$
  - *Augmentation*: If $X \rightarrow Y$, then $XZ \rightarrow YZ$ for any Z
  - *Transitivity*: If $X \rightarrow Y$ and $Y \rightarrow Z$, then $X \rightarrow Z$

- Sound and complete inference rules for FDs!

- Some additional rules (that follow from AA):
  - *Union*: If $X \rightarrow Y$ and $X \rightarrow Z$, then $X \rightarrow YZ$
  - *Decomposition*: If $X \rightarrow YZ$, then $X \rightarrow Y$ and $X \rightarrow Z$
  - See if you can prove these!

Closure of a set of Attributes

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

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

Example:

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

Closures:

$name^+ = \{name, color\}$

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

$color^+ = \{color\}$

Keys

- A **superkey** is a set of attributes A_1, ..., A_n s.t. for any other attribute B, we have A_1, ..., $A_n \rightarrow B$

- A **candidate key** (or sometimes just key) is a minimal superkey
  - A superkey and for which no subset is a superkey

Computing (Super)Keys

- For all sets X, compute X^+
- If X^+ = [all attributes], then X is a superkey
- Try reducing to the minimal X's to get the candidate key

Boyce-Codd Normal Form

A simple condition for removing redundancy/anomalies from relations:

A relation R is in BCNF if and only if:

Whenever there is a nontrivial FD: $A_1A_2\ldots A_n \rightarrow B$,
then $A_1A_2\ldots A_n$ is a super-key for R.

- Non-trivial means RHS is not a subset of LHS
- "Whenever a set of attributes of R is determining another attribute, it should determine <u>all</u> attributes of R."

Why does this make sense?

Say R(A, B, C) with AB as the key has an FD: A→C.
Then C is being repeated for multiple Bs

Example

Name	SSN	Phone Number
Jia	123-32-9931	(201) 555-1234
Jia	123-32-9931	(206) 572-4312
Marco	909-43-4486	(908) 464-0028
Marco	909-43-4486	(212) 555-4000

What are the dependencies?

SSN ➔ Name

Is the left side a superkey?

No

Is it in BCNF?

No.

Decompose it into BCNF

SSN	Name
123-32-9931	Jia
909-43-4486	Marco

SSN	Phone Number
123-32-9931	(201) 555-1234
123-32-9931	(206) 572-4312
909-43-4486	(908) 464-0028
909-43-4486	(212) 555-4000

SSN ➔ Name

Now is it in BCNF?

Summary

- ER diagrams
- ER diagrams → relations
- Functional dependencies
- Keys
- BCNF