Outline

- Announcements
  - Anyone still not registered??

- E/R diagrams
  - Sec. 4.1- 4.4 [Old edition: Chapter 2]

- From E/R diagrams to relations
  - Sec. 4.5 and 4.6 [Old edition: Sec. 3.2 and 3.3]
Database Design

- Why do we need it?
  - Need a way to model real world entities in terms of relations
  - Not easy to go from real-world entities to a database schema

- Consider issues such as:
  - What entities to model
  - How entities are related
  - What constraints exist in the domain
  - How to achieve good designs

- Several formalisms exist
  - We discuss E/R diagrams
Database Design Process

**Today**
- Data Modeling
- E/R diagrams

**Next lecture**
- Refinement
- Relations
- Conceptual Schema
- SQL Tables
- Physical Schema
- Files

Next lecture

![Diagram showing the database design process](Image)
Conceptual Schema Design

Conceptual Model:

Relational Model: plus FD’s
(FD = Functional Dependency)

Normalization: Eliminates anomalies
Entity / Relationship Diagrams

Objects → entities
Classes → entity sets

Attributes are like in ODL
(ODL = Object Definition Language)

Relationships: like in ODL except
- first class citizens (not associated with classes)
- not necessarily binary

Product
entity set

All entities in the same entity set have the same attributes

A relationship may have attributes too!

Association between 2 or more entities

buy

address
Keys in E/R Diagrams

- Every entity set must have a key

![E/R Diagram: Product with attributes: name, price, category]
What is a Relationship?

- A mathematical definition:
  - if A, B are sets, then a relationship R is a subset of $A \times B$

- A={$1,2,3$}, B={$a,b,c,d$},
  - $A \times B = \{(1,a),(1,b), \ldots, (3,d)\}$
  - $R = \{(1,a), (1,c), (3,b)\}$

- makes is a subset of Product $\times$ Company:
Multiplicity of E/R Relations

- one-one:
  ![Diagram of one-one relationship]

- many-one:
  ![Diagram of many-one relationship]

- many-many:
  ![Diagram of many-many relationship]
What does this mean?
Multi-way Relationships

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

![Diagram showing a multi-way relationship between Product, Purchase, Person, and Store]
Q: What does the arrow mean?

A: A given person buys a given product from at most one store
Key Constraints in Multi-way Relationships

**Q:** What does the arrow mean?

**A:** A given person buys a given product from at most one store AND every store sells to every person at most one product.
Key Constraints in Multi-way Relationships

**Q:** How do we say that every person shops at at most one store?

**A:** Cannot. This is the best approximation. (Why only approximation?)
Converting Multi-way Relationships to Binary

- ProductOf
- StoreOf
- BuyerOf

Date

Purchase

Product

Store

Person
Design Principles

What’s wrong?

Moral: be faithful to the specifications of the app!
Design Principles

What’s wrong?

Moral: pick the right kind of entities!
Design Principles

What’s wrong?

Moral: don’t complicate life more than necessary!
From E/R Diagrams to Relational Schema

- Entity set $\rightarrow$ relation
- Relationship $\rightarrow$ relation
Product Set to Relation

Product(name, category, price)

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadgets</td>
<td>$19.99</td>
</tr>
</tbody>
</table>
Relationships to Relations

- **price**
- **category**
- **name**
- **startYear**
- **stockprice**
- **name**

**Product** — makes — **Company**

Watch out for attribute name conflicts

**Makes**\( (\text{product-name}, \text{product-category}, \text{company-name}, \text{year}) \)

<table>
<thead>
<tr>
<th>ProductName</th>
<th>ProductCategory</th>
<th>CompanyName</th>
<th>startYear</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadgets</td>
<td>GizmoWorks</td>
<td>1963</td>
</tr>
</tbody>
</table>

Foreign keys
Relationships to Relations (with constraints)

Makes(prodName, category, companyName, year)

Only keep Product keys as primary key

Better solution: get rid of Makes, modify Product:

<table>
<thead>
<tr>
<th>prodName</th>
<th>Category</th>
<th>Price</th>
<th>startYear</th>
<th>CompanyName</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadgets</td>
<td>$19.99</td>
<td>1963</td>
<td>GizmoWorks</td>
</tr>
</tbody>
</table>
Multi-way Relationships to Relations

- **Product**: price, name
- **Store**: name, address
- **Person**: ssn, name

**Purchase**(prodName, storeName, ssn)
Some objects in a class may be special
  Define a new class
  Better: define a subclass

So --- we define subclasses in E/R
Subclasses

Product

- name
- price
- category

Software Product

- isa
- platforms

Educational Product

- isa
- ageGroup

isa

isa

isa

isa

isa
Subclasses to Relations

**Product**

<table>
<thead>
<tr>
<th>Name</th>
<th>Price</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>99</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>49</td>
<td>photo</td>
</tr>
<tr>
<td>Toy</td>
<td>39</td>
<td>gadget</td>
</tr>
</tbody>
</table>

**Software Product**

<table>
<thead>
<tr>
<th>Name</th>
<th>platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>unix</td>
</tr>
</tbody>
</table>

**Educational Product**

<table>
<thead>
<tr>
<th>Name</th>
<th>ageGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>todler</td>
</tr>
<tr>
<td>Toy</td>
<td>retired</td>
</tr>
</tbody>
</table>
E/R Inheritance

Entity sets overlap

No need for multiple inheritance
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

- Solution 1: acceptable, but imperfect (why?)
Modeling Union Types with Subclasses

- Solution 2: better, more laborious

In fact there is no formal way to represent disjoint vs overlapping subclasses. The Ramakrishnan book suggests writing it down.
Constraints in E/R Diagrams

- 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.

- **Other constraints:** peoples’ ages are between 0 and 150.
Keys in E/R Diagrams

No formal way to specify multiple keys
Single Value Constraints

makes

v. s.

makes
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?
Entity sets are weak when their key comes from classes to which they’re related.
Handling Weak Entity Sets

We should have a rounded arrow here, otherwise Dependents key would have NULL value!

Employee(ssn, name, dept)
Dependents(ssn, name, age)

No need to represent policy separately