Practice and Applications of Data Management
CMPSCI 345

Lecture 08: ER Model
Important note

- Small differences from textbook to textbook

345
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 exists
  - We discuss E/R diagrams
Database Design Process

Today

Data Modeling → Refinement → SQL Tables → Files

- E/R diagrams
- Relations
- Conceptual Schema
- Physical Schema
Conceptual Schema Design

Conceptual Model:

Relational Model: plus FDs
(FD = Functional Dependency)

Normalization: Eliminates anomalies
Entity / Relationship Diagrams

- **Product**
  - entity set

- **address**
  - All entities in the same entity set have the same attributes

- **buys**
  - Association between 2 or more entities
  - A relationship may have attributes too!
Keys in E/R Diagrams

- Every entity set must have a key
What is a Relationship?

- 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:

![Diagram](image-url)
Multiplicity of E/R Relations

- one-one:

- many-one

- many-many
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]
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
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
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
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 may have only one biological 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

m. s.
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?
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
Entity 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

Watch out for attribute name conflicts

**Makes** (productName, product-category, companyName, 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>
Relationships to Relations (with constraints)

Make a table of relationships between Product and Company:

<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>

Better solution: get rid of Makes, modify Product:

Make a table of relationships between Product and Company:

<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

Purchase(prodName, storeName, ssn)
Modeling Subclasses

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

```
Products

- Software products
- Educational products
```

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

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

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

<table>
<thead>
<tr>
<th>Educational Product</th>
<th>Name</th>
<th>ageGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>toddler</td>
<td></td>
</tr>
<tr>
<td>Camera</td>
<td>adult</td>
<td></td>
</tr>
</tbody>
</table>
E/R Inheritance

Entity sets overlap

No need for multiple inheritance
Modeling Union Types With Subclasses

FurniturePiece

Person

Company

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 \(\text{(why?)}\)
Modeling Union Types with Subclasses

- Solution 2: better, more laborious

In fact there is no formal way to represent disjoint vs overlapping subclasses.
Weak Entity Sets

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