Entity/Relationship Modelling

Outline

• E/R model (Chapter 5)
• From E/R diagrams to relational schemas (Chapter 5)
• Constraints in SQL (Chapter 4)
1. Database Design

- Modelling
  - Decide which part of reality is represented in a database
  - Agree on structure of the database before deciding on a particular implementation

- Conceptual Modelling
  - Oriented towards application and users
  - Independent of technology and implementation

Some Terminology

- Real world perception
- Classes of similar objects (and their relationships)
- Database
- Instances
- Abstraction / Description
- E/R Diagram
- (relational) Schema
- Type (of table)
  - Student
    - name
    - firstname
    - birthdate
    - section
    - year
Conceptual Modelling

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

Database Design Formalisms

1. Entity/Relationship model (E/R):
   - More relational in nature.
   - Very widely used
2. Object Definition Language (ODL):
   - Closer in spirit to object-oriented models (e.g. Java)
   - Will not be covered

Both can be translated (semi-automatically) to relational schemas
2. Entity / Relationship Diagrams

Objects → entities
Classes → entity sets

Attributes are like in Java.

Relationships: like object references in Java except
- first class citizens (not associated with classes)
- bidirectional

```
Product
  ↓
name
category
price

Product
  ↓
buys

Company
  ↓
name
stockprice
employs

Person
  ↓
address
name
ssn

Person
  ↓
buys
```
Keys in E/R Diagrams

- Every entity set must have a key
  - a key is an attribute that has a different value for every entity

What is a Relation?

- A mathematical definition:
  - if A and 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

What does this say?
Multi-way Relationships

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

Product

Purchase

Store

Person

Arrows in Multiway Relationships

Q: what does the arrow mean?

A: if I know the store, person, invoice, I know the movie too
Q: what do these arrows mean?

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

Q: how do I say: “invoice determines store”?

A: no good way; best approximation:
Roles in Relationships

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

Attributes on Relationships
Converting Multi-way Relationships to Binary

3. 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

**Makes** (product-name, product-category, company-name, year)

<table>
<thead>
<tr>
<th>Product-name</th>
<th>Product-Category</th>
<th>Company-name</th>
<th>Starting-year</th>
</tr>
</thead>
<tbody>
<tr>
<td>gizmo</td>
<td>gadgets</td>
<td>gizmoWorks</td>
<td>1963</td>
</tr>
</tbody>
</table>

(watch out for attribute name conflicts)
No need for StartYear in *Makes*. Modify *Product*:

<table>
<thead>
<tr>
<th>name</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**

```sql
Purchase(prodName, stName, ssn)
```
4. Design Principles

What’s wrong?

- **Product** ➔ **Purchase** ➔ **Person**

- **Country** ➔ **President** ➔ **Person**

Moral: be faithful!

Design Principles: What’s Wrong?

- **Product** ➔ **Purchase** ➔ **Store**
  - **date**
  - **personName**
  - **personAddr**

Moral: pick the right kind of entities.
Design Principles: What’s Wrong?

- **Moral:** don’t complicate life more than it already is.

Design Principles

- **Product**
- **Purchase**
- **Store**
- **Person**
- **Dates**
- **Price**
- **Date**
Information Loss

Moral: avoid "navigation traps"

Modelling Subclasses

Some objects in a class may be special
• define a new class
• better: define a subclass

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

- Think in terms of records:
  - Product
    - field1
    - field2
  - SoftwareProduct
    - field1
    - field2
    - field3
  - EducationalProduct
    - field1
    - field2
    - field4
    - field5
### Subclasses to Relations

![Diagram showing subclass hierarchy](image)

<table>
<thead>
<tr>
<th></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>

### Modelling Subclass Structure

![Diagram showing subclass hierarchy](image)
Modelling 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?)
Modelling Union Types with Subclasses

Solution 2: better, more laborious

5. Constraints in E/R Diagrams

Finding constraints is part of the modelling 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

Underline:

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

Person

- name
- address
- ssn

Product

- name
- category
- price

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?
Weak Entity Sets

Entity sets are weak if their key comes from other classes to which they are related.

Cannot be identified by its own attributes alone.

Handling Weak Entity Sets

Convert to a relational schema (in class)

Team(number,sport,university-name)
6. Constraints in SQL

- A constraint = a property that we’d like our database to hold
- The system will enforce the constraint by taking some actions:
  - forbid an update
  - or perform compensating updates
Keys

CREATE TABLE Product (  
    name CHAR(30) PRIMARY KEY,  
    category VARCHAR(20))

OR:

CREATE TABLE Product (  
    name CHAR(30),  
    category VARCHAR(20),  
    PRIMARY KEY (name))

Keys with Multiple Attributes

CREATE TABLE Product (  
    name CHAR(30),  
    category VARCHAR(20),  
    price INT,  
    PRIMARY KEY (name, category))

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
<th>Price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Gadget</td>
<td>10</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
<td>20</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Photo</td>
<td>30</td>
</tr>
<tr>
<td>Gimp</td>
<td>Gadget</td>
<td>40</td>
</tr>
</tbody>
</table>
Other Keys

CREATE TABLE Product (  
    productID CHAR(10),  
    name CHAR(30),  
    category VARCHAR(20),  
    price INT,  
    PRIMARY KEY (productID),  
    UNIQUE (name, category))

There is at most one PRIMARY KEY; there can be many UNIQUE

Foreign Key Constraints

CREATE TABLE Purchase (  
    prodName CHAR(30) REFERENCES Product(name),  
    date DATETIME)

prodName is a foreign key to Product(name)  
name must be a key in Product
CREATE TABLE Purchase (    prodName CHAR(30),    REFERENCES Product(name), date DATETIME)

<table>
<thead>
<tr>
<th>ProdName</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
</tr>
</tbody>
</table>

**Foreign Key Constraints**

OR

CREATE TABLE Purchase (    prodName CHAR(30),    category VARCHAR(20), date DATETIME,    FOREIGN KEY (prodName, category) REFERENCES Product(name, category)

(name, category) must be a PRIMARY KEY
What happens during updates?

Types of updates:

- In Purchase: insert/update
- In Product: delete/update

<table>
<thead>
<tr>
<th>ProdName</th>
<th>Store</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>Wiz</td>
</tr>
<tr>
<td>Camera</td>
<td>Ritz</td>
</tr>
<tr>
<td>Camera</td>
<td>Wiz</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Name</th>
<th>Category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>Photo</td>
</tr>
<tr>
<td>OneClick</td>
<td>Photo</td>
</tr>
</tbody>
</table>

What happens during updates?

- SQL has three policies for maintaining referential integrity:
- **Reject** violating modifications (default)
- **Cascade**: after a delete/update do a delete/update
- **Set-null** set foreign-key field to NULL
Constraints on Attributes and Tuples

- Constraints on attributes:
  - NOT NULL -- obvious meaning...
  - CHECK condition -- any condition!
- Constraints on tuples
  - CHECK condition

CREATE TABLE Purchase (prodName CHAR(30)
  CHECK (prodName IN
  SELECT Product.name
  FROM Product),
  date DATETIME NOT NULL)

What is the difference from Foreign-Key?
General Assertions

CREATE ASSERTION myAssert CHECK NOT EXISTS(
    SELECT Product.name
    FROM Product, Purchase
    WHERE Product.name = Purchase.prodName
    GROUP BY Product.name
    HAVING count(*) > 200)

Final Comments on Constraints

- Can give them names, and alter later
- We need to understand exactly *when* they are checked
- We need to understand exactly *what* actions are taken if they fail
Summary

• E/R Models to model the real world
• Adhere to certain design recommendations
• Carefully use constraints