Introduction to Data Management
CSE 344

Section 7: E/R Diagrams
Database Design Process

Conceptual Model:

Relational Model:
Tables + constraints
And also functional dep.

Normalization:
Eliminates anomalies

Conceptual Schema

Physical storage details
Physical Schema
Entity / Relationship Diagrams

- Entity set = a class
  - An entity = an object

- Attribute

- Relationship
Multiplicity of E/R Relations

- one-one:
  ![Graph 1](image)

- many-one:
  ![Graph 2](image)

- many-many:
  ![Graph 3](image)
What does this say?
Multi-way Relationships

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

Can still model as a mathematical set (Q. how ?)

A. As a set of triples \( \subseteq \) Person X Product X Store
Q: What does the arrow mean?

A: A given person buys a given product from at most one store.

[Arrow pointing to E means that if we select one entity from each of the other entity sets in the relationship, those entities are related to at most one entity in E]
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.
3. Design Principles

What’s wrong?

Product → Purchase ← Person

Country → President → Person

Moral: Be faithful to the specifications of the application!
Design Principles: What’s Wrong?

Moral: pick the right kind of entities.
From E/R Diagrams to Relational Schema

- Entity set → relation
- Relationship → relation
**Entity Set to Relation**

\[ \text{Product}(\text{prod-ID}, \text{category}, \text{price}) \]

<table>
<thead>
<tr>
<th>prod-ID</th>
<th>category</th>
<th>price</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo55</td>
<td>Camera</td>
<td>99.99</td>
</tr>
<tr>
<td>Pokemn19</td>
<td>Toy</td>
<td>29.99</td>
</tr>
</tbody>
</table>
N-N Relationships to Relations

Represent this in relations
N-N Relationships to Relations

Orders \( \text{prod-ID}, \text{cust-ID}, \text{date} \)
Shipment \( \text{prod-ID}, \text{cust-ID}, \text{name}, \text{date} \)
Shipping-Co \( \text{name}, \text{address} \)

<table>
<thead>
<tr>
<th>prod-ID</th>
<th>cust-ID</th>
<th>name</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo55</td>
<td>Joe12</td>
<td>UPS</td>
<td>4/10/2011</td>
</tr>
<tr>
<td>Gizmo55</td>
<td>Joe12</td>
<td>FEDEX</td>
<td>4/9/2011</td>
</tr>
</tbody>
</table>
N-1 Relationships to Relations

Represent this in relations
N-1 Relationships to Relations

Orders\((prod-ID, cust-ID, date1, name, date2)\)
Shipping-Co\((name, address)\)

Remember: no separate relations for many-one relationship
Product

- prod-ID
- price

Purchase

- name
- address

Store

Person

- ssn
- name

Purchase\((\text{prod-ID, ssn, name})\)
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
Other ways to convert are possible
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

Say: each piece of furniture is owned either by a person or by a company

Solution 1. Acceptable but imperfect (What’s wrong?)
Modeling Union Types with Subclasses

Solution 2: better, more laborious

- Person
- Company
- FurniturePiece
- Owner

isa

ownedBy
Weak Entity Sets

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

Team(sport, number, universityName)
University(name)
What makes good schemas?
Integrity Constraints Motivation

An integrity constraint is a condition specified on a database schema that restricts the data that can be stored in an instance of the database.

• ICs help prevent entry of incorrect information
• How? DBMS enforces integrity constraints
  – Allows only legal database instances (i.e., those that satisfy all constraints) to exist
  – Ensures that all necessary checks are always performed and avoids duplicating the verification logic in each application
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 in E/R diagrams

Underline:

- name
- category
- price
- address
- ssn
- name
Single Value Constraints

makes vs. 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.
Q: What does this mean?
A: A Company entity cannot be connected by relationship to more than 99 Product entities.
Constraints in SQL:

- Keys, foreign keys
- Attribute-level constraints
- Tuple-level constraints
- Global constraints: assertions

- The more complex the constraint, the harder it is to check and to enforce
Key Constraints

**Product(name, category)**

```
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

Product(name, category, price)

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>Gizmo</td>
<td>Gadget</td>
<td>40</td>
</tr>
</tbody>
</table>
### Other Keys

```sql
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**

```sql
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.

Referential integrity constraints

May write just Product if name is PK.
Foreign Key Constraints

• Example with multi-attribute primary key

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

• (name, category) must be a KEY in Product
What happens when data changes?

Types of updates:

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

<table>
<thead>
<tr>
<th>Product</th>
<th>Purchase</th>
</tr>
</thead>
<tbody>
<tr>
<td>Name</td>
<td>ProdName</td>
</tr>
<tr>
<td>Gizmo</td>
<td>Gizmo</td>
</tr>
<tr>
<td>Camera</td>
<td>Camera</td>
</tr>
<tr>
<td>OneClick</td>
<td>Camera</td>
</tr>
</tbody>
</table>
What happens when data changes?

• SQL has three policies for maintaining referential integrity:
  • **NO ACTION** reject violating modifications (default)
  • **CASCADE** after delete/update do delete/update
  • **SET NULL** set foreign-key field to NULL
  • **SET DEFAULT** set foreign-key field to default value
    – need to be declared with column, e.g.,
      CREATE TABLE Product (pid INT DEFAULT 42)
CREATE TABLE Purchase ( 
  prodName CHAR(30),
  category VARCHAR(20),
  date DATETIME,
  FOREIGN KEY (prodName, category) 
  REFERENCES Product(name, category) 
  ON UPDATE CASCADE 
  ON DELETE SET NULL )
Constraints on Attributes and Tuples

• Constraints on attributes:
  - NOT NULL
  - CHECK condition

• Constraints on tuples
  - CHECK condition

  -- obvious meaning...
  -- any condition!
CREATE TABLE R (  
A int NOT NULL,  
B int CHECK (B > 50 and B < 100),  
C varchar(20),  
D int,  
CHECK (C >= 'd' or D > 0))
CREATE TABLE Product(
    productId CHAR(10),
    name CHAR(30),
    category VARCHAR(20),
    price INT CHECK (price > 0),
    PRIMARY KEY (productId),
    UNIQUE (name, category))
CREATE TABLE Purchase (  
  prodName CHAR(30)  
    CHECK (prodName IN  
      (SELECT Product.name  
        FROM Product),  
    date DATETIME NOT NULL)
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)
  )
```

But most DBMSs do not implement assertions
Because it is hard to support them efficiently
Instead, they provide triggers