COMP 430
Intro. to Database Systems
ER Implementation as Tables
ER implementation overview

- **Author**
  - ID
  - name

- **Concession**
  - product_id
  - state_id
  - dealer_id

- **Dealership**
  - ID
  - name

Primary keys = keys of entity sets on a “many” side of relation.

Except, sometimes optimize away this table:
- Common cases of one-to-many
- One-to-one
Arity & participation: Common cases
Many-to-many

CREATE TABLE Product (  
    ID, name  
    ... ...  
);  

CREATE TABLE Purchase (  
    prod_ID, person_ID  
    ... ...  
);  

CREATE TABLE Person (  
    ID, name  
    ... ...  
);
Many-to-many

CREATE TABLE Purchase (
    ...,
    PRIMARY KEY (prod_ID, person_ID),
    FOREIGN KEY (prod_ID) REFERENCES Product (ID),
    FOREIGN KEY (person_ID) REFERENCES Person (ID)
);
N-ary many-many

Junction table’s primary key = combination of primary keys of all n tables.
One-to-many / many-to-one

CREATE TABLE Publisher (  
    ...  
    PRIMARY KEY (ID)  
);

CREATE TABLE Book (  
    ...  
    PRIMARY KEY (ISBN),  
    FOREIGN KEY (pub_ID) REFERENCES Publisher (ID)  
);
One-to-many / many-to-one

Way to remember where the additional field is placed:
- **Book** has one publisher, so it can be a field.
- **Publisher** has many books, so it can’t be a field.
One-to-many with total participation

Company

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>address</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Product

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>co_ID</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

CREATE TABLE Company (  
  ID INT PRIMARY KEY
);  
CREATE TABLE Product (  
  ID INT PRIMARY KEY,  
  co_ID VARCHAR(50) NOT NULL,  
  FOREIGN KEY (co_ID) REFERENCES Company (ID)  
);  
Related item must have a value.
One-to-many with weak entity

CREATE TABLE University (  
    name,  
    ...  
)  
    PRIMARY KEY (name)  
);

CREATE TABLE Team (  
    univ_name,  
    ID,  
    sport  
    ...  
)  
    PRIMARY KEY (univ_name, ID),  
    FOREIGN KEY (univ_name) REFERENCES University (name)  
);

Related item must have a value.
Activity: Implement three similar ideas

Each results in a **Purchase** junction table. But what primary keys does it have?
One-to-many with total participation

Concession

<table>
<thead>
<tr>
<th>p_id</th>
<th>state_id</th>
<th>d_id</th>
<th>date</th>
</tr>
</thead>
<tbody>
<tr>
<td>…</td>
<td>…</td>
<td>…</td>
<td>…</td>
</tr>
</tbody>
</table>

CREATE TABLE Concession (  
    ...  
    PRIMARY KEY (p_id, state_id),  
    FOREIGN KEY (p_id) REFERENCES Product (p_id),  
    FOREIGN KEY (state_id) REFERENCES State (state_id),  
    FOREIGN KEY (d_id) REFERENCES Dealer (d_id)  
);
One-to-one with same key

Usually indicates a poor design. Combine into one entity set.

<table>
<thead>
<tr>
<th>ID</th>
<th>name</th>
<th>address</th>
<th>points</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
One-to-one with different keys

Sometimes indicates a poor design.

CREATE TABLE Faculty (  
  ...  
  PRIMARY KEY (emp_ID),  
  FOREIGN KEY (dept_ID) REFERENCES Dept (dept_ID)  
);  

CREATE TABLE Department (  
  ...  
  PRIMARY KEY (dept_ID),  
  FOREIGN KEY (chair_ID) REFERENCES Faculty (emp_ID)  
);  

Faculty

<table>
<thead>
<tr>
<th>emp_ID</th>
<th>name</th>
<th>office</th>
<th>dept_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Dept

<table>
<thead>
<tr>
<th>dept_ID</th>
<th>name</th>
<th>chair_id</th>
</tr>
</thead>
<tbody>
<tr>
<td>...</td>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>
Parallel one-to-many/one relationships

Same idea as before, for each relationship.

CREATE TABLE Person (  
    ...  
    PRIMARY KEY (ID)  
);  

CREATE TABLE Vehicle (  
    ...  
    PRIMARY KEY (ID),  
    FOREIGN KEY (owner_ID) REFERENCES Person (owner_ID),  
    FOREIGN KEY (mechanic_ID) REFERENCES Person (owner_ID)  
);
Arity & participation: Some less common cases
Many-to-many with total participation

CREATE TABLE Purchase (  ...  PRIMARY KEY (prod_ID, person_ID),  FOREIGN KEY (prod_ID) REFERENCES Product (ID),  FOREIGN KEY (person_ID) REFERENCES Person (ID) );

Can enforce with a CHECK constraint. See later.
Many-to-many with total participation

Classic chicken-and-egg problem:

No Product can be created without a related Person.
No Person can be created without a related Product.
One-to-many with total participation

CREATE TABLE Company (  
  ...  
  PRIMARY KEY (ID)  
);

CREATE TABLE Makes (  
  ...  
  PRIMARY KEY (co_ID),  
  FOREIGN KEY (co_ID) REFERENCES Company (ID),  
  FOREIGN KEY (prod_ID) REFERENCES Product (ID)  
);  

CREATE TABLE Product (  
  ...  
  PRIMARY KEY (ID)  
);
Subclasses, superclasses, unions
Subclasses – Three strategies

Dealing with sub- & superclasses is one of relational database’s weaknesses.
Subclasses – Strategy 1

Product(name, price)

SoftwareProduct(name, platforms)
EducationalProduct(name, age_group)

+ Semantically, most accurate representation.

- Typically leads to lots of joins between sub- & superclass
Subclasses – Strategy 2

- Avoids joins between sub- & superclasses.
- Requires disjoint subclasses.
- Poor if superclass is directly related to other entity sets.

Employee

HourlyEmployee (ID, name, rate)
SalaryEmployee (ID, name, salary)
Subclasses – Strategy 3

Product(name, price, platforms, age_group)

Use NULL when attribute not applicable.

+ Avoids joins between sub- & superclasses.
- Joins between subclass & other entity sets now involve more data.
Activity: Implement as Schemas

Publisher
- Publishes
  - Book
    - ISBN
    - title
    - num_copies
    - is_self_pub
    - num_copies

Chapter
- PartOf
  - Writing
    - ID
    - title

Author
- Writes
  - Writing
    - ID
    - name

Book
- GoesOnLoan
- Borrows

Loan

Borrower
- card_num
- address
- name