Relational Database Design
Part II
CPS 196.3
Introduction to Database Systems

E/R model: review
- Entity sets
  - Keys
  - Weak entity sets
- Relationship sets
  - Attributes on relationships
  - Multiplicity
  - Roles
  - Binary versus N-ary relationships
    - Modeling N-ary relationships with weak entity sets and binary relationships
    - ISA relationships

Database design steps: review
- Understand the real-world domain being modeled
- Specify it using a database design model (e.g., E/R)
- Translate specification to the data model of DBMS (e.g., relational)
- Create DBMS schema

Next: translating an E/R design to a relational schema
Translating entity sets

- An entity set translates directly to a table
  - Attributes → columns
  - Key attributes → key columns

```
Student (SID, name)
```
```
Course (CID, title)
```

Translating weak entity sets

- Remember the “borrowed” key attributes
- Watch out for attribute name conflicts

```
Building (building_name, year)
```
```
Rooms (building_name, room_number, capacity)
```
```
Seats (building_name, room_number, seat_number, left_or_right)
```

Translating relationship sets

- A relationship set translates to a table
  - Keys of connected entity sets → columns
  - Attributes of the relationship set (if any) → columns
  - Multiplicity of the relationship set determines the key of the table

```
Enroll (SID, CID, grade)
```
Translating double diamonds

- Recall that a double-diamond relationship set connects a weak entity set to another entity set.
- No need to translate because the relationship is implicit in the weak entity set’s translation.

Translating subclasses & ISA (approach 1)

- Entity-in-all-superclasses approach
  - An entity is represented in the table for each subclass to which it belongs.
  - A table include only the attributes attached to the corresponding entity set, plus the inherited key.
Translating subclasses & ISA (approach 2)

- Entity-in-most-specific-class approach
  - An entity is only represented in one table (corresponding to the most specific entity set to which the entity belongs)
  - A table includes the attributes attached to the corresponding entity set, plus all inherited attributes

```
<table>
<thead>
<tr>
<th>Table</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students</td>
<td>name, office</td>
</tr>
<tr>
<td>Course</td>
<td>title</td>
</tr>
<tr>
<td>Enroll</td>
<td>SID, CID, title</td>
</tr>
<tr>
<td>GradStudent</td>
<td>SID, name, office</td>
</tr>
</tbody>
</table>
```

```
\{444, "Apu", "D444"\} ∈ GradStudent (SID, name, office)
```

Translating subclasses & ISA (approach 3)

- All-entities-in-one-table approach
  - One relation for the root entity set, with all attributes found anywhere in the network of subclasses
  - Use a special NULL value in columns that are not relevant for a particular entity

```
<table>
<thead>
<tr>
<th>Table</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Students</td>
<td>name, office</td>
</tr>
<tr>
<td>Course</td>
<td>title</td>
</tr>
<tr>
<td>Enroll</td>
<td>SID, CID, title</td>
</tr>
<tr>
<td>GradStudent</td>
<td>SID, name, office</td>
</tr>
</tbody>
</table>
```

```
\{444, "Apu", NULL\} ∈ GradStudent (SID, name, office)
```

Comparison of three approaches

- Entity-in-all-superclasses
  - Student (SID, name), GradStudent (SID, office)
  - Pro:
  - Con:

- Entity-in-most-specific-class
  - Student (SID, name), GradStudent (SID, name, office)
  - Pro:
  - Con:

- All-entities-in-one-table
  - Student (SID, name, office)
  - Pro:
  - Con:
A complete example

Simplifications and refinements

An alternative design

- Encode the type of train/station as a column rather than creating subclasses
- Difficult to enforce some constraints
  - Type must be either “local” or “express”
  - Express trains only stop at express stations
  - Fortunately, they can be expressed/declared explicitly as database constraints in SQL
  - Arguably a better design because it is simpler!
Design principles

- KISS
- Avoid redundancy
- Use your common sense