Announcements (September 8)

- Homework #1 due in 7 days (next Thursday)
- Details of the course project and a list of suggested ideas will be available next Tuesday

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

![Diagrams of entity sets]

Translating weak entity sets

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

![Diagrams of weak entity sets]
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

\[ \text{Enroll (SID, CID, grade)} \]

More examples

\[ \text{Enroll (SID, CID, TID)} \]

\[ \text{Marry (husband\_SSN, wife\_SSN)} \]

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

\[ \text{is subsumed by} \]

\[ \text{Rooms (building\_name, room\_number, capacity)} \]
Translating subclasses & ISA (approach 1)

- **Entity-in-all-superclasses approach ("E/R style")**
  - An entity is represented in the table for each subclass to which it belongs
  - A table includes only the attributes directly attached to the corresponding entity set, plus the inherited key

```
Students
SID
  name
  office
GradStudents
  (444, "Apu") ∈ Student (SID, name)
  (142, "Bart") ∈ Student (SID, name)
  (444, "D444") ∈ GradStudent (SID, office)

Courses
CID
title
Enroll
SID
  name
  office
GradStudents
  (444, "Apu") ∈ GradStudent (SID, office)
```

Translating subclasses & ISA (approach 2)

- **Entity-in-most-specific-class approach ("OO style")**
  - 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

```
Students
SID
  name
  office
GradStudents
  (444, "Apu") ∈ Student (SID, name)
  (142, "Bart") ∈ Student (SID, name)
  (444, "D444") ∈ GradStudent (SID, name, office)

Courses
CID
title
Enroll
SID
  name
  office
GradStudents
  (444, "Apu", "D444") ∈ GradStudent (SID, name, office)
```

Translating subclasses & ISA (approach 3)

- **All-entities-in-one-table approach ("NULL style")**
  - 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

```
Students
SID
  name
  office
GradStudents
  (444, "Apu", "D444") ∈ GradStudent (SID, name, office)
```

```
Courses
CID
title
Enroll
SID
  name
  office
GradStudents
  (444, "Apu", "D444") ∈ GradStudent (SID, name, office)
  (142, "Bart", NULL) ∈ GradStudent (SID, name, office)
  (444, "D444") ∈ GradStudent (SID, 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

- Eliminate LocalTrain table

- Eliminate LocalStation table
An alternative design

- Encode the type of train/station as a column rather than creating subclasses
- Some constraints are no longer captured
  - 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
  - Keep It Simple, Stupid
- Avoid redundancy
  - Redundancy wastes space, complicates updates and deletes, promotes inconsistency
- Capture essential constraints, but don’t introduce unnecessary restrictions
- Use your common sense
  - Warning: Mechanical translation procedures given in this lecture are no substitute for your own judgment