Relational Database Design
Part II
CPS 116
Introduction to Database Systems

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

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

Translating weak entity sets
- Remember the “borrowed” key attributes
- Watch out for attribute name conflicts

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

Translate:

```
Students (SID, name)
Courses (CID, title)
Enroll (SID, CID, grade)
```

Next:

```
Relating

```

Translate:

```
Rooms (building_name, room_number, capacity)
Buildings (name, year)
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

```
Students  Enroll  Course
SID name
grade

Enroll (SID, CID, grade)
```

More examples

```
More examples

- Students  Enroll  Courses
  - (SID, name)  Enroll  (CID, title)

- Persons  Marry
  - (husband SSN, wife SSN)  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

```
Rooms  Buildings
number
capacity

RoomInBuilding (room building name, room number, building name)

is subsumed by
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  Enroll  Course
SID name

Enroll (SID, CID)
```

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  Enroll  Course
SID name
office

GradStudents
444, "Apu", "D444"
```

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  Enroll  Course
SID name
office

GradStudents
444, "Apu", "D444"
```

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  Enroll  Course
SID name
office

GradStudents
444, "Apu", "D444"
```
Comparison of three approaches

- **Entity-in-all-superclasses**
  - Student (SID, name), GradStudent (SID, office)
  - **Pro:** All students are found in one table
  - **Con:** Attributes of grad students are scattered in different tables

- **Entity-in-most-specific-class**
  - Student (SID, name), GradStudent (SID, name, office)
  - **Pro:** All attributes of grad students are found in one table
  - **Con:** Students are scattered in different tables

- **All-entities-in-one-table**
  - Student (SID, name), GradStudent (SID, name, office)
  - **Pro:** Everything is in one table
  - **Con:** Too many NULL's; complicated if class hierarchy is complex

Simplifications and refinements

- **Eliminate LocalTrain table**
  - Can be computed as \( \pi_{\text{number}}(\text{Train}) \rightarrow \text{ExpressTrain} \)
  - Slightly harder to check that local_train_number is indeed a local train number

- **Eliminate LocalStation table**
  - It can be computed as \( \pi_{\text{number}}(\text{Station}) \rightarrow \text{ExpressStation} \)

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