Relational Database Design
Part II

CompSci 316
Introduction to Database Systems

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 E/R design to relational schema

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

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

Announcements (Tue. Sep. 10)

- Homework #1 is due in a week
- Project is now officially “assigned”
  - See “Assignments” on course website for handout
  - Let me know if there is anything I can do to help you form project ideas/groups
- Interesting project ideas in upcoming talk
  - Derek Willis from *The New York Times*
  - 11:45am-12:45pm, Monday Oct. 7; Perkins LINK 5
- Anonymous feedback is welcome
  - See course website for link
- Interested in Piazza?
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 (SID, C#)
```

More examples

```
SID

name

Student

Enroll (SID, C#)

TA's

name

grade

Courses

Enroll (SID, C#, TID)
```

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 (SID, name)

GradStudent

Office

444, "Apu", "D444"

(444, "Apu") ∈ Student (SID, name)

Enroll (SID, C#, TID)

444, "Apu", "D444" ∈ 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

Enroll (SID, C#)

GradStudent

Office

444, "Apu", "D444"

(444, "Apu") ∈ Student (SID, name)

Enroll (SID, C#, TID)

(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 in the network of subclasses (plus a "type" attribute when needed)
  - Use a special NULL value in columns that are not relevant for a particular entity

```
Students

Enroll (SID, C#)

GradStudent

Office

444, "Apu", "D444"

(444, "Apu") ∈ Student (SID, name, office)

Enroll (SID, C#, TID)

(444, "Apu", "D444") ∈ GradStudent (SID, name, office)
```
Comparison of three approaches

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

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

- All-entities-in-one-table
  - Student (SID, type, name, office)
  - Pros: Everything is in one table
  - Cons: Too many NULL’s, complicated if class hierarchy is complex

Simplifications and refinements

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

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

An alternative design

- Encode the type of train/station as a column rather than creating subclasses
- What about the following constraints?
  - Type must be either “local” or “express”
  - Express trains only stop at express stations
  - They can be expressed/declared explicitly as database constraints in SQL (as we will see later in course)
- Arguably a better design because it is simpler!

Design principles

- KISS
  - Keep It Simple, Stupid
- Avoid redundancy
  - Redundancy wastes space, complicates modifications, 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