Relational Database Design
Part II

CPS 116
Introduction to Database Systems

Announcements (September 7)

- Homework #1 assigned today
  - Due on September 19
  - Start early!!!
  - Help session next week (to be scheduled via email)
- "Notes" vs. "final" versions of lecture slides
- Handout box outside my office
- 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 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
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

<table>
<thead>
<tr>
<th>SID</th>
<th>name</th>
</tr>
</thead>
</table>

Courses

<table>
<thead>
<tr>
<th>CID</th>
<th>title</th>
</tr>
</thead>
</table>

Enroll (SID, CID, grade)

More examples

Students

<table>
<thead>
<tr>
<th>SID</th>
<th>name</th>
</tr>
</thead>
</table>

Courses

<table>
<thead>
<tr>
<th>CID</th>
<th>title</th>
</tr>
</thead>
</table>

Enroll (SID, CID, TID)

Persons

<table>
<thead>
<tr>
<th>SSN</th>
<th>husband</th>
<th>wife</th>
</tr>
</thead>
</table>

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 in Building

<table>
<thead>
<tr>
<th>room_number</th>
<th>room_capacity</th>
</tr>
</thead>
</table>

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

<table>
<thead>
<tr>
<th>Entities</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>SID, name</td>
</tr>
<tr>
<td>Enroll</td>
<td>SID, CID</td>
</tr>
<tr>
<td>Course</td>
<td>CID, title</td>
</tr>
<tr>
<td>GradStudent</td>
<td>SID, office</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Entities</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>SID, name, office</td>
</tr>
<tr>
<td>Enroll</td>
<td>SID, CID</td>
</tr>
<tr>
<td>Course</td>
<td>CID, title</td>
</tr>
<tr>
<td>GradStudent</td>
<td>SID, name, office</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>Entities</th>
<th>Attributes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Student</td>
<td>SID, name, office</td>
</tr>
<tr>
<td>Enroll</td>
<td>SID, CID</td>
</tr>
<tr>
<td>Course</td>
<td>CID, title</td>
</tr>
<tr>
<td>GradStudent</td>
<td>SID, name, office</td>
</tr>
</tbody>
</table>

(444, "Apu", "D444", NULL) ∈ GradStudent (SID, name, office)

(142, "Bart", NULL) ∈ Student (SID, name)

(142, "Bart") ∈ Student (SID, name)

(444, "Apu") ∈ Student (SID, name)

(444, "D444") ∈ GradStudent (SID, name, office)

(444, "Apu") ∈ Student (SID, name)

(142, "Bart") ∈ Student (SID, name)

(444, "D444") ∈ GradStudent (SID, name, office)

(444, "Apu") ∈ Student (SID, name)

(142, "Bart", NULL) ∈ Student (SID, name)
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

![ER diagram showing relationships between entities such as Trains, Stations, LocalTrainStops, ExpressTrainStops, LocalTrains, LocalStations, ExpressTrains, ExpressStations, LocalTrainStops, and ExpressTrainStops.]

Simplifications and refinements

- Eliminate LocalTrain table
- Eliminate LocalStation table
An alternative design

Train (train_number, engineer, type)
Station (name, address, type)
TrainStop (train_number, station_name, time)

✈️ 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

✈️ 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