Relational Database Design
Part II

CPS 116
Introduction to Database Systems

Announcements

- Homework #1 due in 7 days (Thursday, Sept. 9)
- Details of the course project and presentations will be available next Tuesday
- Discussion session next week: Homework #1 Q&A
  - Vote on meeting time in this lecture

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

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 \( \rightarrow \) columns
  - Attributes of the relationship set (if any) \( \rightarrow \) columns
  - Multiplicity of the relationship set determines the key of the table

\[
\begin{array}{c}
\text{Students} \quad \text{Enroll} \quad \text{Courses} \\
\text{name} \quad \text{grade} \quad \text{CID} \quad \text{title} \\
\end{array}
\]

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

\[
\begin{array}{c}
\text{Rooms} \quad \text{Buildings} \quad \text{name} \quad \text{year} \\
\text{capacity} \quad \text{building name} \quad \text{room number} \quad \text{building name} \quad \text{room number} \quad \text{capacity} \\
\end{array}
\]

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 include only the attributes attached to the corresponding entity set, plus the inherited key

\[
\begin{array}{c}
\text{Students} \quad \text{Enroll} \quad \text{Courses} \\
\text{name} \quad \text{office} \quad \text{CID} \quad \text{title} \\
\end{array}
\]

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

\[
\begin{array}{c}
\text{Students} \quad \text{Enroll} \quad \text{GradStudents} \\
\text{name} \quad \text{office} \quad \text{SID} \quad \text{name} \quad \text{office} \\
\end{array}
\]

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

\[
\begin{array}{c}
\text{Students} \quad \text{Enroll} \quad \text{Courses} \\
\text{name} \quad \text{office} \quad \text{SID} \quad \text{CID} \\
\end{array}
\]
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:** All students are found in one table
  - **Con:** Too many NULL’s; complicated if class hierarchy is complex

A complete example

An alternative design

Simplifications and refinements

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

- **Eliminate LocalStation table**
  - It can be computed as $\pi_{\text{name}}(\text{Station})$ — ExpressStation

Design principles

- **KISS**
  - Keep It Simple, Stupid!
- **Avoid redundancy**
  - Redundancy wastes space, complicates updates, and promotes inconsistency
- **Use your common sense**
  - Warning: Mechanical translation procedures given in this lecture are no substitute for your own judgment