Announcements (Tue. Sep. 13)

- Homework #1 due in one week
  - You should have started by now
- Course project description available!
  - Choice of “standard” or “open”
  - Team of size 1-4, but 1- and 4-person teams need approval from me
  - Two milestones + demo/report
  - Milestone #1 due in one month, right after fall break

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

\[
\begin{align*}
\text{Students} & \quad \text{Enroll} \quad \text{Course} \\
\text{name} & \quad \text{grade} \quad \text{title} \\
\end{align*}
\]

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

More examples

\[
\begin{align*}
\text{Students} & \quad \text{Enroll} \quad \text{Courses} \\
\text{name} & \quad \text{grade} \\
\end{align*}
\]

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

\[
\begin{align*}
\text{Persons} & \quad \text{Mentor} \\
\text{name} & \quad \text{SSN} \\
\end{align*}
\]

\[\text{Mentor (mentor\_SSN, protégé\_SSN)}\]

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

\[
\begin{align*}
\text{Rooms} & \quad \text{Buildings} \\
\text{name} & \quad \text{year} \\
\end{align*}
\]

\[\text{Rooms\_Building (room\_building\_name, room\_number, building\_name)}\]

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.

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.

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.
Comparison of three approaches

- **Entity-in-all-superclasses**
  - Student (SID, name), GradStudent (SID, name, office)
  - Pro:
  - Con:

- **Entity-in-most-specific-class**
  - Student (SID, name), GradStudent (SID, name, office)
  - Pro:
  - Con:

- **All-entities-in-one-table**
  - Student (SID, type, name, office)
  - Pro:
  - Con:

A complete example

```
Trans (number, name, address, number, engineer)
LocalTrain (number, name, address, number)
ExpressTrain (number, name, address, number)
Station (name, address)
LocalStation (name)
ExpressStation (name)
```

Simplifications and refinements

Train (number, engineer), LocalTrain (number), ExpressTrain (number)
Station (name, address), LocalStation (name), ExpressStation (name)

- Eliminate LocalTrain table
  - Redundant: can be computed as $\pi_{\text{number}}(\text{Train})$ — $\text{ExpressTrain}$
  - Why is redundancy bad?
  - 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

| 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**
  - 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