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

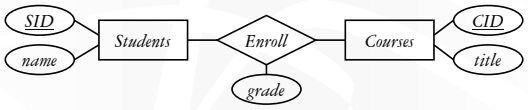
Database design steps: review

- 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

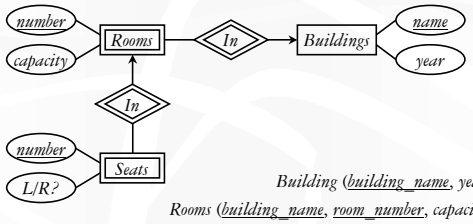


Student (*SID*, *name*)

Course (*CID*, *title*)

Translating weak entity sets

- Remember the "borrowed" key attributes
- Watch out for attribute name conflicts



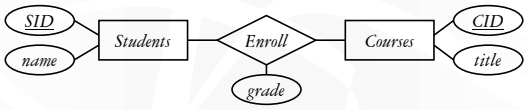
Building (*building_name*, *year*)

Rooms (*building_name*, *room_number*, *capacity*)

Seats (*building_name*, *room_number*, *seat_number*, *left_or_right*)

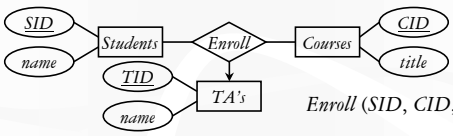
Translating relationship sets

- 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

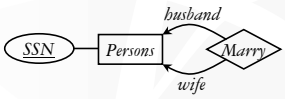


Enroll (*SID*, *CID*, *grade*)

More examples

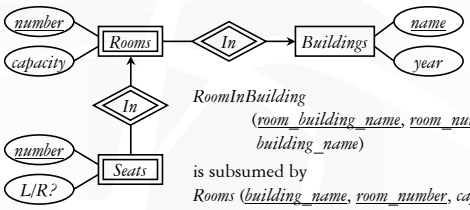


Enroll (*SID*, *CID*, *TID*)



Translating double diamonds

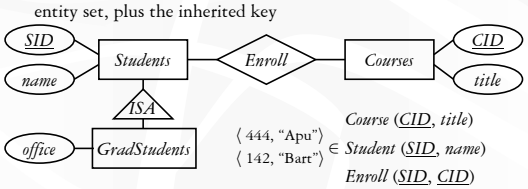
- Recall that a double-diamond relationship set connects a weak entity set to another entity set
- No need to translate because the relationship is implicit in the weak entity set's translation



RoomInBuilding
(*room_building_name*, *room_number*, *building_name*)
is subsumed by
Rooms (*building_name*, *room_number*, *capacity*)

Translating subclasses & ISA (approach 1)

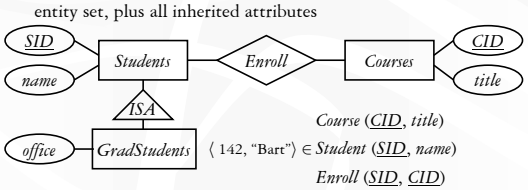
- Entity-in-all-superclasses approach ("E/R style")
  - An entity is represented in the table for each subclass to which it belongs
  - A table include only the attributes attached to the corresponding entity set, plus the inherited key



Course (*CID*, *title*)
⟨ 444, "Apu" ⟩
⟨ 142, "Bart" ⟩ ∈ *Student* (*SID*, *name*)
Enroll (*SID*, *CID*)
⟨ 444, "D444" ⟩ ∈ *GradStudent* (*SID*, *office*)

Translating subclasses & ISA (approach 2)

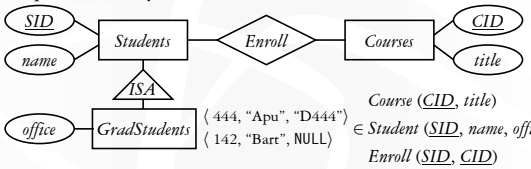
- Entity-in-most-specific-class approach ("OO style")
  - An entity is only represented in one table (corresponding to the most specific entity set to which the entity belongs)
  - A table includes the attributes attached to the corresponding entity set, plus all inherited attributes



Course (*CID*, *title*)
⟨ 142, "Bart" ⟩ ∈ *Student* (*SID*, *name*)
Enroll (*SID*, *CID*)
⟨ 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 anywhere in the network of subclasses
  - Use a special NULL value in columns that are not relevant for a particular entity



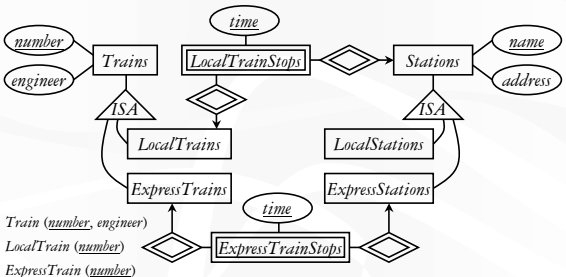
⟨ 444, "Apu", "D444" ⟩
⟨ 142, "Bart", NULL ⟩

Course (*CID*, *title*)
∈ *Student* (*SID*, *name*, *office*)
Enroll (*SID*, *CID*)

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



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

LocalTrainStop (*local_train_number*, *station_name*, *time*)
ExpressTrainStop (*express_train_number*, *express_station_name*, *time*)

Simplifications and refinements

Train (*number*, *engineer*), *LocalTrain* (*number*), *ExpressTrain* (*number*)
Station (*name*, *address*), *LocalStation* (*name*), *ExpressStation* (*name*)
LocalTrainStop (*local_train_number*, *station_name*, *time*)
ExpressTrainStop (*express_train_number*, *express_station_name*, *time*)

- Eliminate *LocalTrain* table

- Eliminate *LocalStation* table

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 promotes inconsistency
- Use your common sense
  - Warning: Mechanical translation procedures given in this lecture are no substitute for your own judgment