Relational Database Design: E/R-Relational Translation

Introduction to Databases

CompSci 316 Spring 2017

DUKE COMPUTER SCIENCE

Announcements (Wed. Jan 25)

- Project details posted
  - Milestone 1 due: Feb 27
- Project mixer class next Wed (Feb 1)
  - start thinking about cool ideas
  - prepare a few slides
- HW1, Problem2 clarification
  - You need not and should not use division operation or aggregate operations to answer any question (not covered in class)
- A few of you are still not on piazza!
  - you might miss some important discussions and notifications
  - join soon!

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

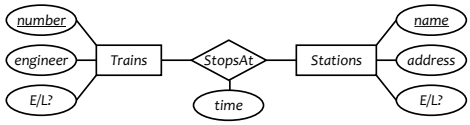
E/R model (Lecture 3): review

- 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

Case study 2 (from Lecture 3)

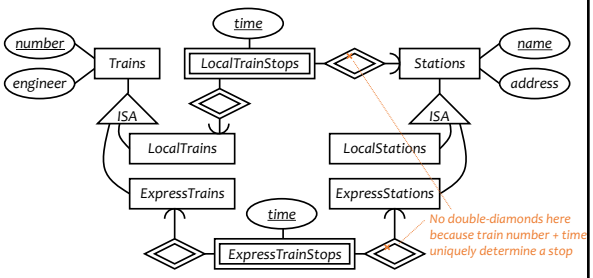
- Design a database consistent with the following:
  - A station has a unique name and an address, and is either an express station or a local station
  - A train has a unique number and an engineer, and is either an express train or a local train
  - A local train can stop at any station
  - An express train only stops at express stations
  - A train can stop at a station for any number of times during a day
  - Train schedules are the same everyday

Case study 2: first design



- What is wrong in this E/R diagram?
- Nothing in this design prevents express trains from stopping at local stations
  - ☞We should capture as many constraints as possible
- A train can stop at a station only once during a day
  - ☞We should not introduce unintended constraints

Case study 2: second design



Is the extra complexity worth it?
-- we will come back to this point

Database design steps: review

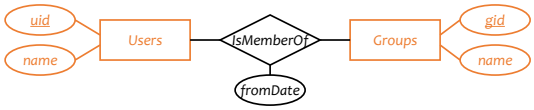
- Understand the real-world domain being modeled
- Specify it using a database design model (e.g., E/R)
- Translate specification to the data model of DBMS (e.g., relational)
- Create DBMS schema

☞Today: translating E/R design to relational schema

☞Next in Lecture 5: how to remove unwanted redundancy by "normalization" from this initial design

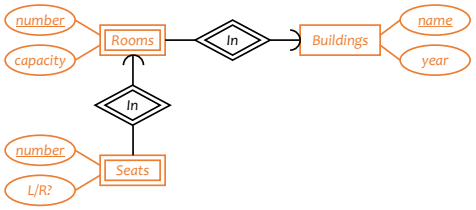
Translating entity sets

- An entity set translates directly to a table
  - Attributes → columns
  - Key attributes → key columns



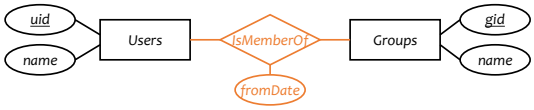
Translating weak entity sets

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



Translating relationship sets

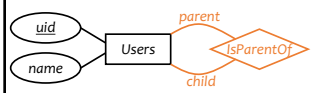
- A relationship set translates to a table
  - Keys of connected entity sets → columns
  - Attributes of the relationship set (if any) → columns
  - Multiplicity of the relationship set determines the key of the table

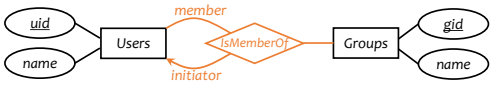


If we have an arrow to "Groups", what should be a key?

More examples

Also identify Keys

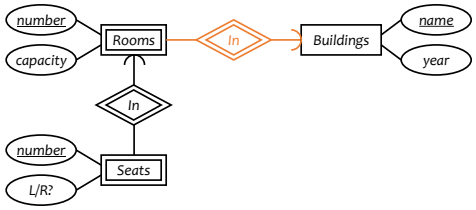




13

Translating double diamonds?

- Recall that a double-diamond (supporting) 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

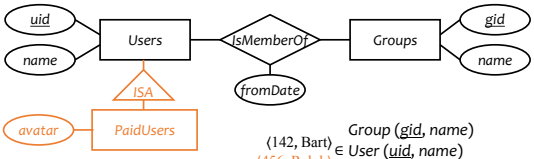


What if a double diamond has its own attribute?

14

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

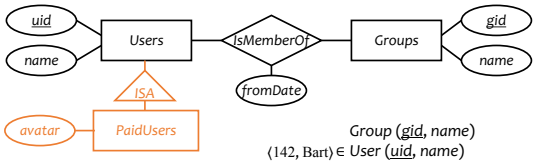


⟨142, Bart⟩ ∈ *Group* (*gid*, *name*)
⟨456, Ralph⟩ *User* (*uid*, *name*)
Member (*uid*, *gid*, *from_date*)
⟨456, ☺⟩ ∈ *PaidUser* (*uid*, *avatar*)

15

Translating subclasses & ISA: approach 2

- Entity-in-most-specific-class approach ("OO style")
  - An entity is only represented in one table (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

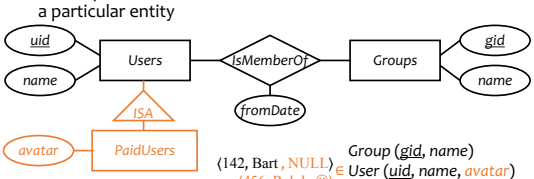


Group (*gid*, *name*)
⟨142, Bart⟩ ∈ *User* (*uid*, *name*)
Member (*uid*, *gid*, *from_date*)
⟨456, Ralph, ☺⟩ ∈ *PaidUser* (*uid*, *name*, *avatar*)

16

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



⟨142, Bart , NULL⟩ ∈ *Group* (*gid*, *name*)
⟨456, Ralph, ☺⟩ *User* (*uid*, *name*, *avatar*)
Member (*uid*, *gid*, *from_date*)

"NULL" = unknown – frequently used in DBMS

17

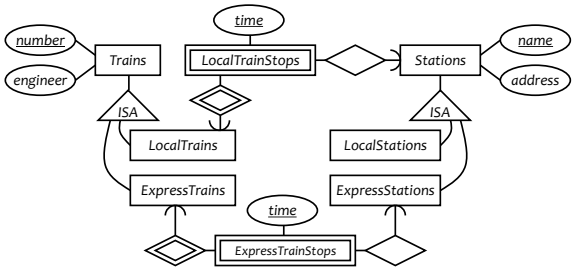
Comparison of three approaches

- Entity-in-all-superclasses
  - *User* (*uid*, *name*), *PaidUser* (*uid*, *avatar*)
  - Pro:
  - Con:
- Entity-in-most-specific-class
  - *User* (*uid*, *name*), *PaidUser* (*uid*, *name*, *avatar*)
  - Pro:
  - Con:
- All-entities-in-one-table
  - *User* (*uid*, [*type*,]*name*, *avatar*)
  - Pro:
  - Con:

which one requires least space?

18

A complete example



19

Simplifications and refinements

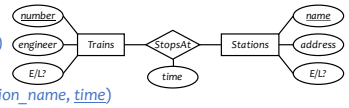
Train (*number*, *engineer*), *LocalTrain* (*number*), *ExpressTrain* (*number*)
Station (*name*, *address*), *LocalStation* (*name*), *ExpressStation* (*name*)
LocalTrainStop (*local_train_number*, *station_name*, *time*)
ExpressTrainStop (*express_train_number*, *express_station_name*, *time*)

- What else can be eliminated (can be computed from other tables)?

20

An alternative design (first one!)

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
- What about the following constraints?
  - Type must be either "local" or "express"
  - Express trains only stop at express stations

21

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

http://ungenius.files.wordpress.com/2010/03/thehomer.jpg

22

Next: VM Lab
by Yuhao and Junyang