Relational Database Design: E/R-Relational Translation

Announcements (Thu. Jan. 30)
• HW2/Lab1 due tonight (Thurs, Jan 30, 11:59 pm)
• HW3 Q1-Q3 posted
  • Q4, extra credit to be posted next week after the material is covered in class
  • Many parts, start early!
• Please form your groups by next Thursday Feb 6!
  • So that we can help you find a group if needed well before MS1 is due
  • Project formation spreadsheet shared
  • 5 members for standard projects please! (otherwise we may have to shuffling later, better if you do it yourself)
  • If you want to do an open project, let me know asap

Announcements – contd. (Thu. Jan. 30)
• HW extension requests (see the course policy)
  • We cannot accommodate requests for “I need more time” to be fair to all
  • For unforeseen situations not in our control like medical reasons, you must submit an Incapacitation form and copy your academic dean while requesting an extension and mention the extra time you need (typically 1-2 days)
  • Make sure that you have an email from me accepting the extension request and specifying the deadline.
  • That deadline is final for you and late submissions with penalty do not apply
  • Be careful as the next hw would be posted

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
You designed an ER digram
Translate it to a Relational Database
Train (number, engineer, type)
Station (name, address, type)
TrainStop (train_number, station_name, time)

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

```
Users gid name
Groups gid name
```

Translating weak entity sets

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

```
Rooms
  capacity
  number
  L/R?

Buildings
  name
  year

Seats
  number
  L/R?
```

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

```
Users gid name
Groups gid name
```

More examples

```
Users
  member
  initiator

Groups
  parent
  child

Parent (parent_uid, child_uid)
```

Translating double diamonds?

```
Rooms
  capacity
  number
  L/R?

Buildings
  name
  year

Seats
  number
  L/R?
```

Translating subclasses & ISA

```
Users
  avatar

PaidUsers
```

```
Groups
```

```
Users
  ISA
```

```
Groups
```

```
Users
  ISA
```

```
Groups
```

```
Users
  ISA
```

```
Groups
```

```
Users
  ISA
```

```
Groups
```
Translating subclasses & ISA: approach 1

- Entity-in-all-superclasses approach (“E/R style”)

Translating subclasses & ISA: approach 2

- Entity-in-most-specific-class approach (“OO style”)

Translating subclasses & ISA: approach 3

- All-entities-in-one-table approach (“NULL style”)

Comparison of three approaches

A complete example

- 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

- Draw the ER diagram and translate into relational model

A complete example: ER diagram

Next, Relational Design!
A complete example: ER diagram

- Trains
  - number
  - engineer
- Stations
  - name
  - address
- ExpressTrains
- LocalTrains
- ExpressStations
- LocalStations

Simplifications and refinements

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

- Arguably a better design because it is simpler!

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

Quick clarifications: RA questions

- What is “some”?
  - At least one
  - e.g., Drinkers frequent some bars that serve beer X

- What is “only”?
  - E.g., Drinkers frequent only bars that serve beer X

- What is “every”?
  - E.g., Drinkers frequent every bars that serve beer X