transforming OMs to RDBs

Daniel Jackson
two approaches

- Database schema
  - In normal form? (Yes -> done)
  - No -> fix schema by adding tables

Normalization approach

- Transformation approach
  - Object model
    - Transform
      - Database schema
object model for reviews

exercise: draw an OM
› users write reviews for subjects, with ratings and content
› users have emails, passwords, first and last names
› users write reviews on particular date
› subjects are grouped into categories
an object model

- User
  - Email
  - Password
  - by
  - content
  - on
  - date
- Review
  - rating
  - about
- Subject
  - category
  - name
  - first, last
- Name
- Category
  - name
  - last
  - first
- Text
  - email
  - password
  - !
designing a schema

exercise: design a schema for this OM

your design is a set of tables
› each table has a name and a set of column names
› each column has a type (integer, string, bool)
### step 1: one table/entity

<table>
<thead>
<tr>
<th>entity</th>
<th>attribute</th>
</tr>
</thead>
<tbody>
<tr>
<td>has identity, persists over time</td>
<td>no identity distinct from value</td>
</tr>
<tr>
<td>represents mutable, composite object</td>
<td>represents immutable, primitive object</td>
</tr>
</tbody>
</table>

#### Diagram:
- **User**
  - Email
  - Password
  - by
  - password
- **Review**
  - content
  - on
  - rating
- **Subject**
  - name
  - category
- **Name**
- **Category**
- **Text**
- **Date**
result of identifying tables

- users()
- subjects()
- reviews()
- categories()
step 2: assign relations to columns

assign by multiplicity
› put relation in entity at many end
› so about goes in Review, not Subject
result of assigning relations

- users (email, password)
- subjects (name, category)
- reviews (about, rating, content, on, by)
- categories (name)
step 3: deal with options

want to avoid nulls
› so what if multiplicity is “?”

option (a): lowering
› create a subset in the OM
› now handle as a generalization

option (b): create special value
› eg, define rating=0 for no rating
step 4: define keys

identify keys for each entity
› users (email, password)
› subjects (name, category)
› reviews (about, rating, content, on, by)
› categories (name)

choose a primary key or add surrogate
› users (email), reviews (id), subjects(id)

identify foreign keys
› reviews (about) -> subjects(id)

in practice, often best to use surrogate keys
› with autoincrement
converting to SQL

so reviews table is now

```sql
create table if not exists reviews (id, about, rating, content, on, by)
```