transforming OMs to RDBs

Daniel Jackson
two approaches

- **database schema**
  - in normal form? (yes or no)
    - yes: done
    - no: fix schema by adding tables
      - normalization approach

- **object model**
  - transform
    - database schema
      - transformation approach
a sample object model

Date ! on Entry by User

Subject ! about Review content Text

Name ! category

Category

Email

Password

Rating
**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:
- **Date** ! on **Entry** by **User**
- **Email** !
- **Password** ?
- **Subject** ! about **Review** content !
- **Name** ! name ?
- **Category** !
- **Rating** ?
- **Text** !
result of creating tables

› entries ()
› 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

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

want to avoid nulls

so if multiplicity is ?

› either lower (see OM pattern)
› or create special value

in this case

› `users.rating` is optional
› so define `rating=0` for no rating
step 4: define keys

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

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

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

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

```
create table if not exists reviews (
  id integer primary key,
  on integer,
  by integer references users(id),
  content text,
  rating integer,
  about integer references subjects(id),
  unique (about, by)
);
```