Introduction to Data Management

Lecture #5
(E-R → Relational, Cont’d.)

Instructor: Mike Carey
mjcarey@ics.uci.edu

Announcements

- HW#2 is underway...!
  - Based on our provided HW #1 solution
- Today’s plan:
  - Final chapter of E-R → Relational translation
  - More (hopefully) clarifying examples and Q&A
  - Start on relational DB design theory (if time)
- This week’s discussion sessions will...
  - Start out with a quick E-R → relational quiz
  - Move on to Q&A about HW #1’s solution
  - Also perhaps MySQL installation Q&A
**ISA Mapping Revisited**

- As in Java or other PLs, attributes are inherited
- If we declare A ISA B, then every A entity is also considered to be a B entity instance.

- **Overlap constraints:** Can Joe be both an Hourly_Emps as well as a Contract_Emps instance? (Allowed/disallowed)
- **Covering constraints:** Must every Employees entity also be either an Hourly_Emps or a Contract_Emps entity? (Yes/no)

**ISA Hierarchy Translation Options**

- **I. “Delta table” approach:**
  - $\text{Emps}(\text{ssn}, \text{name}, \text{lot})$ ← (All Emps partly reside here)
  - $\text{Hourly}_\text{Emps}(\text{ssn}, \text{wages, hrs}\_\text{worked})$
  - $\text{Contract}_\text{Emps}(\text{ssn, contractid})$

- **II. “Union of tables” approach:**
  - $\text{Emps}(\text{ssn, name, lot})$
  - $\text{Hourly}_\text{Emps}(\text{ssn, name, lot, wages, hrs}\_\text{worked})$
  - $\text{Contract}_\text{Emps}(\text{ssn, name, lot, contractid})$

- **III. “Mashup table” approach:**
  - $\text{Emps}(\text{kind, ssn, name, lot, wages, hrs}\_\text{worked, contractid})$
ISA Considerations (cont’d.)

- **Query convenience**
  - Ex: List the names of all Emps in lot 12A

- **PK enforcement**
  - Ex: Make sure that ssn is unique for all Emps

- **Relationship targets**
  - Ex: Lawyers table REFERENCES Contract_Emps

- **Handling of overlap constraints**
  - Ex: Sally is under a contract for her hourly work

- **Space and query performance tradeoffs**
  - Ex: List all the info about hourly employee 123
  - Ex: What if most employees are “just plain employees”?

Mapping Advanced ER Features

- **Multi-valued (vs. single-valued) attributes**
  - Employees
    - Employees_phones(ssn, phone)
      - ssn is an FK in this table
      - (ssn, phone) is its PK

- **Derived (vs. base/stored) attributes**

- **Composite (vs. atomic) attributes**
  - Employees
    - (ssn, name, address_snum, address_street, address_city, address_zip)
SQL Views (and Security)

- A **view** is just a relation, but we store its **definition** rather than storing the (materialized) set of tuples.

  
  ```sql
  CREATE VIEW YoungActiveStudents (name, grade)
  AS SELECT S.name, E.grade
  FROM Students S, Enrolled E
  WHERE S.sid = E.sid and S.age < 21
  ```

- Views can be used to present needed information while hiding details of underlying table(s).
  - Given YoungStudents (but not Students or Enrolled), we can see (young) students $S$ who have are enrolled but not see the $cid$’s of their courses.

SQL Views (Cont’d.)

- Other view uses in our ER translation context might include:
  - Derived attributes, e.g., age (vs. birthdate)
  - Simplifying/eliminating join paths (for SQL)
  - Beautifying the “Mashup table” approach (to ISA)

  ```sql
  CREATE VIEW EmployeeView (ssn, name, bdate, age)
  AS SELECT E.ssn, E.name, E.bdate,
          TIMESTAMPDIFF(YEAR, E.bdate, CURDATE())
  FROM Employees E
  ```
Another Mapping Example: Binary vs. Ternary Relationships

(“Better design”)

- The key constraints let us combine Purchaser with Policies and Beneficiary with Dependents.
- Participation constraints lead to NOT NULL constraints.

(Note: Primary key attributes are all NOT NULL as well – check documentation to see if that’s implicit or explicit!

```
CREATE TABLE Policies (  
policyid INTEGER,  
cost REAL,  
emp_ssn CHAR(11) NOT NULL,  
PRIMARY KEY (policyid),  
FOREIGN KEY (emp_ssn) REFERENCES Employees  
ON DELETE CASCADE)
```

```
CREATE TABLE Dependents (  
pname CHAR(20),  
age INTEGER,  
policyid INTEGER) NOT NULL,  
PRIMARY KEY (pname, policyid),  
FOREIGN KEY (policyid) REFERENCES Policies  
ON DELETE CASCADE)
```
Review: Binary vs. Ternary Relationships

CREATE TABLE Employees (ssn CHAR(11), name CHAR(20), lot INTEGER, PRIMARY KEY (ssn))

CREATE TABLE Policies (policyid INTEGER, cost REAL, emp_ssn CHAR(11) NOT NULL, PRIMARY KEY (policyid), FOREIGN KEY (emp_ssn) REFERENCES Employees ON DELETE CASCADE)

CREATE TABLE Dependents (pname CHAR(20), age INTEGER, policyid INTEGER NOT NULL, PRIMARY KEY (pname, policyid), FOREIGN KEY (policyid) REFERENCES Policies ON DELETE CASCADE)

Review: Putting The Basics Together

CREATE TABLE LineItem (lno INTEGER, price REAL, qty INTEGER, PRIMARY KEY (lno))

CREATE TABLE Product (pname CHAR(20), color CHAR(10), listprice REAL, PRIMARY KEY (pname))

CREATE TABLE Order (oid INTEGER, shipto CHAR(20), total REAL, PRIMARY KEY (oid))

CREATE TABLE Customer (cid INTEGER, login CHAR(10), PRIMARY KEY (cid))

CREATE TABLE Placed (oid INTEGER, cid INTEGER, PRIMARY KEY (oid, cid))

CREATE TABLE Lineltem (oid INTEGER, pname CHAR(20), price REAL, qty INTEGER, PRIMARY KEY (oid, lno))

CREATE TABLE Has (oid INTEGER, lno INTEGER, PRIMARY KEY (oid, lno))

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke
Review: Putting It Together (Cont’d.)

CREATE TABLE Customer(
cid INTEGER,
cname VARCHAR(50),
login VARCHAR(20) NOT NULL,
PRIMARY KEY (cid),
UNIQUE (login))

CREATE TABLE Product(
sku INTEGER,
pname VARCHAR(100),
color VARCHAR(20),
listprice DECIMAL(8,2),
PRIMARY KEY (sku))

CREATE TABLE Order(
oid INTEGER,
custid INTEGER,
shipto VARCHAR(200),
total DECIMAL(8,2),
FOREIGN KEY (oid) REFERENCES Product)

CREATE TABLE LineItem(
oid INTEGER,
lno INTEGER,
price DECIMAL(8,2),
qty INTEGER,
sku INTEGER,
FOREIGN KEY (oid, lno) REFERENCES Order
ON DELETE CASCADE,
FOREIGN KEY (sku) REFERENCES Product)

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke

Review: Putting It Together (Cont’d.)

<table>
<thead>
<tr>
<th>Customer</th>
<th>cid</th>
<th>cname</th>
<th>login</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>Smith, James</td>
<td><a href="mailto:jsmith@aol.com">jsmith@aol.com</a></td>
</tr>
<tr>
<td>2</td>
<td>2</td>
<td>White, Susan</td>
<td><a href="mailto:suzie@gmail.com">suzie@gmail.com</a></td>
</tr>
<tr>
<td>3</td>
<td>3</td>
<td>Smith, James</td>
<td><a href="mailto:js@hotmail.com">js@hotmail.com</a></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Product</th>
<th>sku</th>
<th>pname</th>
<th>color</th>
<th>listprice</th>
</tr>
</thead>
<tbody>
<tr>
<td>123</td>
<td>123</td>
<td>Frozen DVD</td>
<td>null</td>
<td>24.95</td>
</tr>
<tr>
<td>456</td>
<td>456</td>
<td>Graco Twin Stroller</td>
<td>green</td>
<td>199.99</td>
</tr>
<tr>
<td>789</td>
<td>789</td>
<td>Moen Kitchen Sink</td>
<td>black</td>
<td>350.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Order</th>
<th>oid</th>
<th>custid</th>
<th>shipto</th>
<th>total</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>3</td>
<td>J. Smith, 1 Main St., USA</td>
<td>199.95</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>1</td>
<td>Mrs. Smith, 3 State St., USA</td>
<td>300.00</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>LineItem</th>
<th>oid</th>
<th>lno</th>
<th>price</th>
<th>qty</th>
<th>item</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>1</td>
<td>1</td>
<td>169.95</td>
<td>1</td>
<td>456</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>15</td>
<td>123</td>
<td>2</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>300.00</td>
<td>789</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Relational Model and E-R Schema Translation: Summary

- Relational model: a tabular representation of data.
- Simple and intuitive, also widely used.
- Integrity constraints can be specified by the DBA based on application semantics. DBMS then checks for violations.
  - Two important ICs: Primary and foreign keys (PKs, FKs).
  - In addition, we always have domain constraints.
- Powerful and natural query languages exist (soon!)
- Rules to translate E-R to relational model
  - Can be done by a human, or automatically (using a tool)