It’s time again for....

Friday Nights
With Databases

Brought to you by...
Today’s Reminders

- Continue to follow the course wiki page
  - http://www.ics.uci.edu/~cs122a/
- Continue to live by the Piazza page
  - https://piazza.com/uci/spring2018/cs122a/home
- First HW assignment is due today
  - Up to 24 hours to finish with a 20% late penalty
- Next HW assignment is available now
  - Translate E-R PEEEza schema into relational form
  - Use our solution schema (out tomorrow at 5pm)

Review: Participation Constraints

- Does every department have a manager?
  - If so, this is a participation constraint: the participation of Departments in Manages is said to be total (vs. partial).
  - Every did value in Departments table must appear in a row of the Manages table (with a non-null ssn value!!)

Database Management Systems 3ed, R. Ramakrishnan and J. Gehrke
Participation Constraints in SQL

- We can capture participation constraints involving one entity set in a binary relationship, but little else (without resorting to the use of triggers).

```
CREATE TABLE Department2 (  
did INTEGER,  
dname CHAR(20),  
budget REAL,  
mgr_ssn CHAR(11) NOT NULL,  
mgr_since DATE,  
PRIMARY KEY (did),  
FOREIGN KEY (mgr_ssn) REFERENCES Employees,  
ON DELETE NO ACTION) (*or: RESTRICT)
```

Review: Weak Entities

- A weak entity can be identified (uniquely) only by considering the primary key of another (owner) entity.
  - Owner entity set and weak entity set must participate in a one-to-many relationship set (1 owner, many weak entities).
  - Weak entity set must have total participation in this identifying relationship set.
**Translating Weak Entity Sets**

- Weak entity set and identifying relationship set are translated into a **single table**.
  - When the owner entity is deleted, all of its owned weak entities must also be deleted.

```sql
CREATE TABLE Dependents2 (
  pname CHAR(20),
  age INTEGER,
  cost REAL,
  ssn CHAR(11) NOT NULL,
  PRIMARY KEY (pname, ssn),
  FOREIGN KEY (ssn) REFERENCES Employees,
  ON DELETE CASCADE)
```

**Review: ISA Hierarchies**

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

- **Overlap constraints:** Can employee Joe be an Hourly_Emps as well as a Contract_Emps entity? *(Allowed/disallowed)*
- **Covering constraints:** Must each Employees entity be either an Hourly_Emps or a Contract_Emps entity? *(Yes/no)*
From ISA Hierarchies to Relations

- **Most general and “clean” approach** (recommended):
  - 3 relations: Employees, Hourly_Emps, and Contract_Emps.
    - **Hourly_Emps**: Every employee recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (hourly_wages, hours_worked, ssn); delete Hourly_Emps tuple if referenced Employees tuple is deleted.
    - Queries about all employees easy; those involving just Hourly_Emps require a join to access the extra attributes.

- **Another alternative**: Hourly_Emps and Contract_Emps.
  - Ex: Hourly_Emps(ssn, name, lot, hourly_wages, hours_worked)
  - If each employee must be in one of the two subclasses...
    - (Q: Can we always do this, then? A: Not w/o redundancy!)

ISA Hierarchy Translation Options

- I. “Delta table” approach (recommended):
  - Emps(ssn, name, lot) ← (All Emps partly reside here)
  - Hourly_Emps(ssn, wages, hrs_worked)
  - Contract_Emps(ssn, contractid)

- II. “Union of tables” approach:
  - Emps(ssn, name, lot)
  - Hourly_Emps(ssn, name, lot, wages, hrs_worked)
  - Contract_Emps(ssn, name, lot, contractid)

- III. “Mashup table” approach:
  - Emps(kind, ssn, name, lot, wages, hrs_worked, contractid)

Things to consider:
- Expected queries?
- PK/unique constraints?
- Relationships/FKs?
- Overlap constraints?
- Space/time tradeoffs?
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**

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

- **Composite (vs. atomic) attributes**

Employees(ssn, name, address_snum, address_street, address_city, address_zip)

EmployeesPhones(ssn, phone)

Employees(name, snum, street, city, zip)

Employees(name, bdate)
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

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 Order (  
  cid CHAR(11),  
  login CHAR(20),  
  oid CHAR(20),  
  shipto CHAR(20),  
  total REAL,  
  PRIMARY KEY (oid))

CREATE TABLE Customer (  
  cid CHAR(11),  
  login CHAR(20),  
  PRIMARY KEY (cid))

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

CREATE TABLE Product (  
  sku INTEGER,  
  fname CHAR(20),  
  color CHAR(20),  
  listprice REAL,  
  PRIMARY KEY (sku))

CREATE TABLE Lineitem (  
  lno CHAR(20),  
  price REAL,  
  qty INTEGER,  
  PRIMARY KEY (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),
    PRIMARY KEY (oid),
    FOREIGN KEY (custid) REFERENCES Customer)

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

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

| Customer | | | |
|----------|-----|----------------|
| cid | cname | login |
| 1 | Smith, James | jsmith@aol.com |
| 2 | White, Susan | suzie@gmail.com |
| 3 | Smith, James | js@hotmail.com |

| Product | | | | |
|---------|-----|-----|------|
| sku | pname | color | listprice |
| 123 | Frozen DVD | null | 24.95 |
| 456 | Graco Twin Stroller | green | 199.99 |
| 789 | Moen Kitchen Sink | black | 350.00 |

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

| LineItem | | | | |
|----------|-----|-----|-----|
| oid | lno | price | qty | item |
| 1 | 1 | 169.95 | 1 | 456 |
| 1 | 2 | 15.00 | 2 | 123 |
| 2 | 1 | 300.00 | 1 | 789 |
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)