CS122A: Introduction to Data Management

Lecture #5
(E-R \(\rightarrow\) Relational, Cont.)

Instructor: Chen Li
Logical DB Design: ER to Relational (Review)

- Entity sets to tables:

```sql
CREATE TABLE Employees
(ssn CHAR(11),
 name CHAR(20),
 lot INTEGER,
 PRIMARY KEY (ssn))
```
In translating a relationship set to a relation, attributes of the relation must include:

- Keys for each participating entity set (as foreign keys).
  - Note: This set of attributes forms a superkey for the relation.
- All descriptive attributes.

```
CREATE TABLE Works_In(
    ssn CHAR(11),
    did INTEGER,
    since DATE,
    PRIMARY KEY (ssn, did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (did) REFERENCES Departments
)
```
Key Constraints (Review)

- Each dept has at most one manager, according to the *key constraint* on Manages.
Translating ER Diagrams with Key Constraints

- Map the relationship to a table (Manages):
  - Note that did is the key now!
  - Separate tables for Employees and Departments.

- But, since each department has a unique manager, we could choose to fold Manages right into Departments.

(Q: Why do that...?)

```
CREATE TABLE Manages (  
  ssn CHAR(11),  
  did INTEGER,  
  since DATE,  
  PRIMARY KEY (did),  
  FOREIGN KEY (ssn) REFERENCES Employees,  
  FOREIGN KEY (did) REFERENCES Departments)
```

```
CREATE TABLE Departments2 (  
  did INTEGER,  
  dname CHAR(20),  
  budget REAL,  
  mgr_ssn CHAR(11),  
  mgr_since DATE,  
  PRIMARY KEY (did),  
  FOREIGN KEY (mgr_ssn) REFERENCES Employees)
```

Note: The relationship info has been pushed to the N-side’s entity table!
Properly Reflecting Key Constraints

- So what are the translated relationship table keys (etc.) when...
  - FooBar is M:N? → FooBar(fooId, barId, baz)
  - FooBar is N:1? → FooBar(fooId, barId, baz)
  - FooBar is 1:N? → FooBar(fooId, barId, baz)
  - FooBar is 1:1? → FooBar(fooId, barId, baz)  (Note: unique)
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 Schema Diagram]

- Employees
  - ssn
  - name
  - lot

- Manages
  - since
  - did

- Departments
  - dname
  - budget

- Works_In
  - since
**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).

```sql
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)
```
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.

![Entity-Relationship Diagram](diagram.png)
Translating Weak Entity Sets

- Weak entity set and identifying relationship set are translated into a single table.
  - When the owner entity is deleted, all 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 Joe be an Hourly_Emps as well as a Contract_Emps entity? *(Allowed/disallowed)*
- **Covering constraints**: Does every Employees entity also have to be an Hourly_Emps or a Contract_Emps entity? *(Yes/no)*
Translating ISA Hierarchies to Relations

- **Most general approach:**
  - 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 get the extra attributes.

- **An alternative:** Hourly_Emps and Contract_Emps.
  - **Hourly_Emps:** ssn, name, lot, hourly_wages, hours_worked.
  - Each employee must be in one of these two subclasses.
    - (Q: Can we always do this, then? A: Not w/o redundancy!)
Review: 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 NOT NULL as well – check documentation to see if that’s implicit or explicit!)*

```sql
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)
```

```sql
CREATE TABLE Dependents (  
    pname    CHAR(20),  
    age      INTEGER,  
    policyid INTEGER,  
    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,  
  PRIMARY KEY (pname, policyid),  
  FOREIGN KEY (policyid)  
  REFERENCES Policies  
  ON DELETE CASCADE)
An Example: Putting It Together

- **Customer**
  - cid
  - cname
  - login

- **Order**
  - oid
  - shipto
  - total
  - Placed

- **LineItem**
  - lno
  - price
  - qty

- **Product**
  - sku
  - pname
  - color
  - listprice

- **Relationships**
  - 1:1 between Customer and Order
  - N:1 between Order and LineItem
  - 1:1 between Product and LineItem

Putting It Together (Cont’d.)

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

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

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)
## Putting It Together (Cont’d.)

### Customer

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

### Product

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

### Order

<table>
<thead>
<tr>
<th>oid</th>
<th>custid</th>
<th>shipto</th>
<th>total</th>
</tr>
</thead>
<tbody>
<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

<table>
<thead>
<tr>
<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>169.95</td>
<td>1</td>
<td>456</td>
</tr>
<tr>
<td>1</td>
<td>2</td>
<td>15.00</td>
<td>2</td>
<td>123</td>
</tr>
<tr>
<td>2</td>
<td>1</td>
<td>300.00</td>
<td>1</td>
<td>789</td>
</tr>
</tbody>
</table>
SQL Views

- 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 dropped using the **DROP VIEW** command.
  - How to handle **DROP TABLE** if there’s a view on the table?
    - DROP TABLE command has options to let the user specify this.
Views and Security

- Views can be used to present necessary information (or a summary) while hiding some details in underlying relation(s).
  - Given YoungStudents, but not Students or Enrolled, we can find students S who have are enrolled, but not the cid’s of the courses they are enrolled in.
Relational Model and E-R Schema Translation: Summary

- 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)