Logical DB Design: ER to Relational

• Entity sets to tables:

CREATE TABLE Employees
(ssn CHAR(11),
name CHAR(20),
lot INTEGER,
PRIMARY KEY (ssn))
Relationship Sets to Tables

• Attributes of the relation must include:
  – Keys for each participating entity set (as foreign keys).
    • 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)
Translating ER Diagrams with Key Constraints

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

- Since each department has a unique manager, we could instead combine Manages and Departments.

```
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 Dept_Mgr(
    did INTEGER,
    dname CHAR(20),
    budget REAL,
    ssn  CHAR(11),
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees)
```
Participation Constraints in SQL

- We can capture participation constraints involving one entity set in a binary relationship, but little else (without resorting to \texttt{CHECK} constraints).

```sql
CREATE TABLE Dept_Mgr(
    did INTEGER,
    dname CHAR(20),
    budget REAL,
    ssn CHAR(11) NOT NULL,
    since DATE,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees,
    ON DELETE NO ACTION)
```
**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 owned weak entities must also be deleted.

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

- As in C++, or other PLs, attributes are inherited.
- If we declare A **ISA** B, 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

**General approach:**

- 3 relations: Employees, Hourly_Emps and Contract_Emps.
  
  - *Hourly_Emps*: Every employee is recorded in Employees. For hourly emps, extra info recorded in Hourly_Emps (hourly_wages, hours_worked, ssn); must delete Hourly_Emps tuple if referenced Employees tuple is deleted).
  
  - Queries involving all employees easy, those involving just Hourly_Emps require a join to get some attributes.

**Alternative: Just Hourly_Emps and Contract_Emps.**

- *Hourly_Emps*: ssn, name, lot, hourly_wages, hours_worked.
  
  - Each employee must be in one of these two subclasses.
Unified Modeling Language

- Standardized general-purpose modeling language for software design
- Based on object-oriented model
- Class diagrams

<table>
<thead>
<tr>
<th>UML</th>
<th>E/R Model</th>
</tr>
</thead>
<tbody>
<tr>
<td>Class</td>
<td>Entity set</td>
</tr>
<tr>
<td>Association</td>
<td>Binary relationship</td>
</tr>
<tr>
<td>Association Class</td>
<td>Attributes on a relationship</td>
</tr>
<tr>
<td>Subclass</td>
<td>Isa hierarchy</td>
</tr>
<tr>
<td>Aggregation</td>
<td>Many-one relationship</td>
</tr>
<tr>
<td>Composition</td>
<td>Many-one relationship with referential integrity</td>
</tr>
</tbody>
</table>
UML Classes

ER Entity Set

Movies

- title
- year
- length
- genre

UML Class

Class name

Methods section typically not used in data modeling

<table>
<thead>
<tr>
<th>Movies</th>
</tr>
</thead>
<tbody>
<tr>
<td>title</td>
</tr>
<tr>
<td>year</td>
</tr>
<tr>
<td>length</td>
</tr>
<tr>
<td>genre</td>
</tr>
</tbody>
</table>

<place for methods>
Associations

Cardinality constraints: one instance of Stars can be connected to at least 0 instance of movies and at most infinite instances of movies.
Referential Integrity

Aggregation: Must by 0..1 (includes 1..1)

Composition: Must be 1..1
Every president runs exactly one studio

Aggregation never named
Association Classes

Stars
- name
- address

0..* Stars-in 0..* Movies

Compensation
- salary
- residuals

Movies
- title
- PK
- year
- PK
- length
- genre
Sub-Class Hierarchies
Modeling Tips

• Faithful to the semantics of the application
• Model only what is needed in the application
• Minimize redundancy (why?)
• Simple is good
• If the model is getting too complicated, take a step back and ask
  – Am i conceptualizing the right entities ?
  – Am i thinking of the right relationships ?
  – Should some relationships become entities ? Vice versa ?
  – Should some attributes become entities ? Vice versa ?