CSC 261/461 – Database Systems
Lecture 8

Fall 2017
• Project 2 is out.

• Not a group project!
Agenda

• More about ER model
• ER model to Relation (Table)
Chen Notation vs Crow’s Foot Notation

Textbook inverts the \((\text{min, max})\) notations. We are following that for consistency.
Special Symbols Used
Strong and Weak Entities

• A weak entity is an entity whose instances cannot exist in the database without the existence of an instance of another entity.

• Any entity that is not weak entity is called a strong entity.
  – Instances of a strong entity can exist in the database independently.

• The weak entity’s identifier is a combination of the identifier of the owner entity and the partial key of the weak entity.
• Double diamond for *supporting* many-one relationship.
• Double rectangle for the weak entity set.
Example: Weak Entity Set

- **name** is almost a key for football players, but there might be two with the same name.
- **number** is certainly not a key, since players on two teams could have the same number.
- But **number**, together with the team **name** related to the player should be unique.
Partial vs Total Participation

• An entity set may participate in a relation either totally or partially.
• **Total participation** means that every entity in the set is involved in the relationship
  – depicted as a double line.

• **Partial participation** means that not all entities in the set are involved in the relationship, e.g., not every professor guides a student
  – depicted by a single line.
Weak Entity, Total Participation and Partial Key

Different Diagrams Representing the Concepts:

1. Diagram 1: Title
   - Title
   - Book ID
   - Book
   - Contains
   - Chapter
   - Chapter ID
HAS-A Relationships

• The relationships in the previous slides are called HAS-A relationships.

• The term is used because each entity instance has a relationship to a second entity instance:
  – An employee has a locker
  – A locker has an employee

• There are also IS-A relationships.
Modeling Subclasses

• Some objects in a class may be special, i.e. worthy of their own class
  • Define a new class?
    • \textit{But what if we want to maintain connection to current class?}
  • Better: define a \textit{subclass}
    • \textit{Ex:}

![Diagram showing subclasses in E/R](attachment: subclasses_diagram.png)

We can define \textit{subclasses} in E/R!
Modeling Subclasses

Child subclasses contain all the attributes of all of their parent classes plus the new attributes shown attached to them in the E/R diagram.
Understanding Subclasses

• Think in terms of records; ex:

  – Product
    - name
    - price

  – SoftwareProduct
    - name
    - price
    - platforms

  – EducationalProduct
    - name
    - price
    - ageGroup

Child subclasses contain all the attributes of all of their parent classes plus the new attributes shown attached to them in the E/R diagram.
Think like tables...

<table>
<thead>
<tr>
<th>name</th>
<th>price</th>
<th>category</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>99</td>
<td>gadget</td>
</tr>
<tr>
<td>Camera</td>
<td>49</td>
<td>photo</td>
</tr>
<tr>
<td>Toy</td>
<td>39</td>
<td>gadget</td>
</tr>
</tbody>
</table>

Software Product

<table>
<thead>
<tr>
<th>name</th>
<th>platforms</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>unix</td>
</tr>
</tbody>
</table>

Educational Product

<table>
<thead>
<tr>
<th>name</th>
<th>ageGroup</th>
</tr>
</thead>
<tbody>
<tr>
<td>Gizmo</td>
<td>toddler</td>
</tr>
<tr>
<td>Toy</td>
<td>retired</td>
</tr>
</tbody>
</table>
• Summary
ACTIVITIES
DRAW AN E/R DIAGRAM FOR FOOTBALL

Use the following simplified model of a football season (concepts to include are underlined):

**Teams** play each other in **Games**. Each pair of teams can play each other multiple times

**Players** belong to **Teams**

A Game is made up of **Plays** that result in a yardage gain/loss, and potentially a touchdown

A Play will contain either a **Pass** from one player to another, or a **Run** by one player
Note that various ER diagrams could work, not just the following one!
Teams play each other in Games. Each pair of teams can play each other multiple times.

Note two copies of the Teams entity here!
Players belong to Teams (assume no trades / changes)
A Game is made up of Plays that result in a yardage gain/loss, and potentially a touchdown
A Play will contain either a **Pass** from one player to another, or a **Run** by one player.
Note that various ER diagrams could work, not just the following one!
ENHANCE YOUR E/R DIAGRAM!

Also make sure to add **(new concepts underlined):**

A player can only belong to one team, a play can only be in one game, a pass/run..?

Players can achieve a **Personal Record** linked to a specific Game and Play

Players have a **weight** which changes in on vs. off-season
A player can only belong to one team, a play can only be in one game, a pass/run...?
Players can achieve a **Personal Record** linked to a specific Game and Play.
Players might have different weights at different times.

Note: point here is that different players might have different numbers of training / weight phases—hence should represent as new entity!
Note that various ER diagrams could work, not just the following one!

ACTIVITY 3
ADD IN: SUBCLASSES, CONSTRAINTS, AND WEAK ENTITY SETS

Concepts to include / model:

Teams belong to cities- model as **weak entity sets**

Players are either on Offense or Defense, and are of types (QB, RB, WR, TE, K)

All passes are to exactly one player; all runs include a player

Make sure you have designated keys for all our concepts!
Teams belong to cities- model as *weak entity sets*
Players are either on Offense or Defense, and are of types (QB, RB, WR, TE, etc.)
Design Theory (ER model to Relations)
CREATE TABLE Employees (  
  ssn char(11),  
  name varchar(30),  
  lot Integer,  
  PRIMARY KEY (ssn))
Relationship Sets (without Constraints) to Tables

- Employees
  - name
  - ssn
  - lot

- Works_In
  - since

- Departments
  - dname
  - budget

- Locations
  - address
  - capacity
CREATE TABLE Works_in(
    ssn char(11),
    did integer (30),
    address varchar(30),
    since date,
    PRIMARY KEY (ssn, did, address),
    FOREIGN KEY (ssn) REFERENCES Employees,
    FOREIGN KEY (address) REFERENCES Locations,
    FOREIGN KEY (did) REFERENCES Departments,
)
CREATE TABLE Reports_To(
    supervisor_ssn char(11),
    subordinate_ssn char(11),
    PRIMARY KEY (supervisor_ssn, subordinate_ssn),
    FOREIGN KEY (supervisor_ssn )
    REFERENCES Employees(ssn),
    FOREIGN KEY (subordinate_ssn )
    REFERENCES Employees(ssn)
)

Relationship Sets (without Constraints) to Tables

CREATE TABLE Reports_To(
    supervisor_ssn char(11),
    subordinate_ssn char(11),
    PRIMARY KEY (supervisor_ssn, subordinate_ssn),
    FOREIGN KEY (supervisor_ssn )
    REFERENCES Employees(ssn),
    FOREIGN KEY (subordinate_ssn )
    REFERENCES Employees(ssn)
)
CREATE TABLE Manages ( ssn char(11),
                        did integer (30),
                        since date,
                        PRIMARY KEY (did),
                        FOREIGN KEY (ssn) REFERENCES Employees,
                        FOREIGN KEY (did) REFERENCES Departments,
                    )
CREATE TABLE Dept_Mgr(
  did integer (30),
  dname varchar(30),
  budget float(30),
  ssn char(11),
  since date,

  PRIMARY KEY (did),
  FOREIGN KEY (ssn) REFERENCES Employees,
)

CSC 261, Spring 2017, UR
CREATE TABLE Dept_Mgr(
    did integer (30),
    dname varchar(30),
    budget float(30),
    ssn char(11),
    since date,
    PRIMARY KEY (did),
    FOREIGN KEY (ssn) REFERENCES Employees ON DELETE NO ACTION
)
CREATE TABLE Dept_Policy (  pname varchar(30),  age integer,  cost float,  ssn char(11),  
PRIMARY KEY (pname, ssn),  
FOREIGN KEY (ssn) REFERENCES Employees  
ON DELETE CASCADE )
Translating Class Hierarchies

Employees

- name
- ssn
- lot

isa

Hourly_Emps

- Hours_worked
- Hourly_wages

Contract_Emps

- Contract_id
Two options

1. We can map each of the entity sets Employees, Hourly_Emps, and Contract_Emps to a distinct relation.

2. We can create just two relations, corresponding to Hourly_Emps and Contract_Emps

Both have their pros and cons

- Redundancy
- Performance
E/R Summary

- E/R diagrams are a visual syntax that allows technical and non-technical people to talk
  - For conceptual design

- Basic constructs: entity, relationship, and attributes

- A good design is faithful to the constraints of the application, but not overzealous
• One customer can have at max 2 loans. One loan can be given to multiple customers.

What it really means:
– One customer can have \((0,2)\) loans
– One loan can be given to \((1,n)\) customer
– This is a many to many scenario
Acknowledgement

• Some of the slides in this presentation are taken from the slides provided by the authors.
• Many of these slides are taken from cs145 course offered by Stanford University.
• Thanks to YouTube, especially to Dr. Daniel Soper for his useful videos.