# **CS579 Fall 2009**

# Class #8 Homework

# **Charlie Kim**

Problems

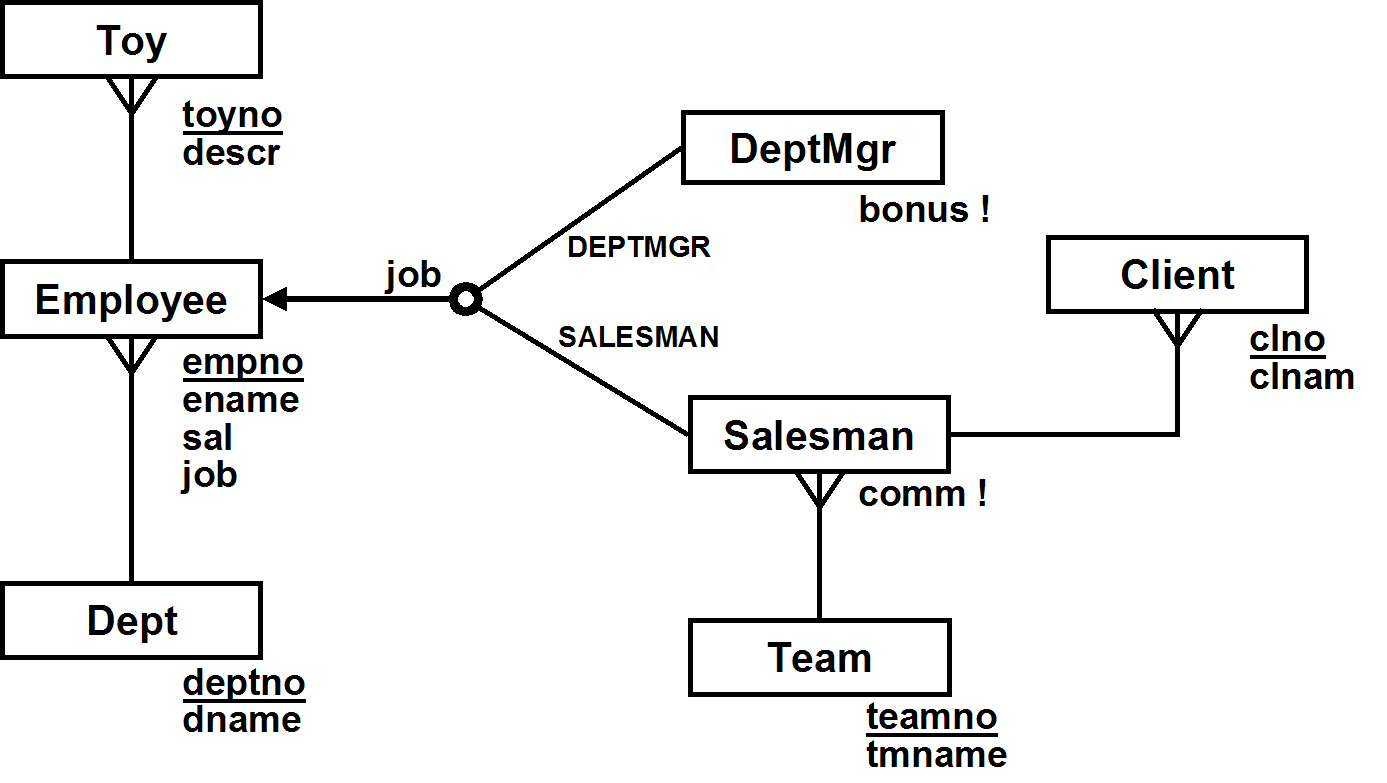
IMPORTANT: You MAY do these problems on your own or with a partner.

If you do any problems with a partner,   
hand in one copy with your names on it.

If you have ANY questions, ***post them on the newsgroup***. Answering questions posted to the newsgroup can positively impact your grade.

**1. Subclasses**

Suppose you are given the following conceptual model:



**a)** Separately draw the Relational Schema resulting from [1] superclass/subclass, [2] superclass-only, *and* [3] subclass-only mappings of the model. Make sure you

* underline primary keys
* make certain that your foreign keys are pointing in the right direction
* indicate cascading deletes when they are required by the conceptual model, but not otherwise

Note: In the **subclass-only** mapping, it would be INCORRECT to have a table which represents *all* employees. However, you will need to add some other table to keep track of employees who are neither salesmen nor dept mgrs.

[1] super-class/subclass

[2] super-class only

[3] subclass only

**b)** **For each of the 3 mappings** (superclass-subclass, superclass-only, and subclass-only), write the (one or possibly more) INSERT statement(s) needed to add a new dept mgr for dept 20 to the database, with empno 5162, ename 'BOAZ', sal 2200, and bonus 200.

**-- superclass-subclass**

**INSERT INTO Employees VALUES ( 5162, ‘BOAZ’, 2200, ‘DEPTMGR’, 20 );**

**INSERT INTO DeptMgrs VALUES ( 5162, 200 );**

**-- superclass only**

**INSERT INTO Employees VALUES ( 5162, ‘BOAZ’, 2200, ‘DEPTMGR’, 20, 200, NULL, NULL );**

**-- subcase only**

**INSERT INTO DeptMgrs VALUES ( 5162, ‘BOAZ’, 2200, 200, 20 );**

**c)** Suppose that we define the compensation of an employee in the following way:

* if they're a dept manager, it is their salary plus their bonus
* if they're a salesman , it is their salary plus their commission
* For all other employees, it is just their salary

**For each of the 3 mappings**, write a single SQL query that lists the name and compensation of each employee, ordered by name (you may assume that all employee have unique names). I recommend that you do them in the order:  
[1] subclass-only, [2] superclass-only, [3] superclass-subclass.

**-- Subclass-only**

**SELECT ename, (sal + bonus) as comp FROM DeptMgrs**

**UNION**

**SELECT ename, (sal + comm) as comp FROM Salesmen**

**UNIOS**

**SELECT ename, sal as comp FROM Staffs;**

**-- superclass-only**

**SELECT ename, ( sal + ( CASE job**

**WHEN ‘DEPTMGR’ THEN bonus**

**WHEN ‘SALESMAN’ THEN comm.**

**ELSE 0 END ) ) as comm FROM Employees;**

**-- superclass/subclass**

**SELECT ename, (sal + bonus) as comp FROM DeptMgrs NATURAL JOIN Employees**

**UNION**

**SELECT ename, (sal + comm) as comp FROM Salesmen NATURAL JOIN Employees**

**UNIOS**

**SELECT ename, sal as comp FROM Eomployees WHERE job NOT IN ( ‘DEPTMGR’, ‘SALESMAN’);**

**2. [Extra Credit] Conceptual Design:**

A tennis club wants to design a database that keeps track of members, with their personal information (name, address, phone number), when they joined, and their member id, and of instructors, with their personal information (name, address, phone number), skill level, and their instructor id.

Courts at the tennis club can be reserved for hourly slots from 8am to 8pm every day, and must be reserved before they can be used, either for games or for lessons. The database keeps track of all reservations, including the one person who reserved each court slot. The database also keeps track of who plays or takes lessons during each court slot.

A court can either be reserved for a singles or doubles game (but only by a member who will play in the game), or for a lesson (by that lesson's instructor). Lessons are either individual lessons (for members only) or group lessons (open to non-members).

The club requires that the identity be recorded of every person who plays on a court (including non-members who play games with members, or who take a group lesson, or instructors). If a non-member plays for the first time, they must provide their personal information (name, address, phone number), and are given a unique id, which they can subsequently use.

There are two parts to this problem (a) and (b)

**a)** Draw the corresponding (Easy) Crow Magnum or UML ER diagram. Show all

* relationship characterizations (not needed for subclass relationships)
* mandatory participation constraints, dependent and identifying relationships as needed.
* all relevant attributes, with primary keys and discriminators denoted appropriately

🡺 You MUST use subclassing. Draw the subclasses correctly. Make sure the arrows point toward the superclass. Correctly distinguish between overlapping and disjoint subclasses. For disjoint subclasses, distinguish between those which are complete and incomplete, and use subclass discriminators where appropriate.

If you use any enumerated attributes (i.e. attributes which take on a small number of fixed values, like TRUE/FALSE, or like BIG/MEDIUM/SMALL), you MUST write a design note which describes the possible values of the attribute and what each value means.

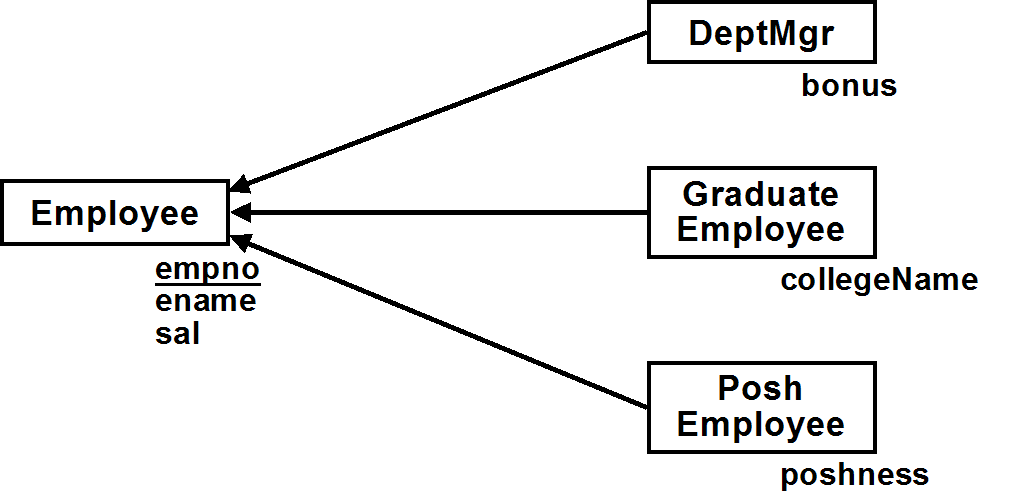
🡺 You MUST use the following entity classes: Lesson, Game, Instructor and Member. You will need additional entity classes as well. In particular, consider classes which are superclasses of the ones above.

**b)** List any **relationship** and **general** conceptual **state** constraints (i.e. NO entity constraints, no pre or post conditions) which are based directly on the requirements, and which are not already redundant with your ER diagram.

**THE APPROACH I PREFER** incorporates more constraints in the ER diagrams (especially by using relationships between subclasses!). In general, this means that fewer or simpler constraints will need to be specified explicitly!

**3. [Hard: EXTRA CREDIT] Overlapping Subclasses**

Suppose you are given the conceptual model described below, which shows three ***overlapping*** subclasses of Employee.



**a)** Draw the Relational Schema or write a brief relational model resulting from a ***subclass-only*** mapping of the conceptual model. Remember that subclass-only means that there is no table (e.g. Emps) which represents all employees. You can only have tables which represent subclasses of employees.

**b)** Based on that relational model, write one or more UPDATE statements to add 100 to the salary of all posh dept managers (i.e. who are both PoshEmployees & DeptMgrs). Note: this is more complex than it might first appear.

IMPORTANT NOTE: An UPDATE statement that uses a view is only legal if it can only update a *single* base table on which the view is based.  
  
ANOTHER IMPORTANT NOTE: This REALLY is more complicated than you might think… (Even if you think you already figured out why it's complicated, it’s probably even trickier than that!)

**c)** Suppose that you use a superclass/subclass mapping, with the tables Emps, Managers, GradEmps and PoshEmps. Add a *single* attribute to Employee (something like a subclass discriminator), so that queries that are based on checking an employee's subclasses can be done by only accessing the Emps table.

(1) First, describe the required attribute and how the values it can take on should be interpreted

(2) Second, write the SQL for the example queries below, showing how they only need to access the Emps table

(a) List the names of all employees who are dept managers  
(b) List the names of all employees who are graduates

(c) List the names of all employees who are posh dept managers

(d) List the names of employees who are neither graduates, posh, nor dept managers

HINT: Look at the last section of lab exercise #1.