# **CS579 Fall 2009**

# Class #8 Homework

# Hardcopy only (no e-mail) due at start of Class #9

# 🡺 E-mail a zip file of your project folder by 11pm on the night *before* the start of Class #9

# Bring the requested hardcopy to class (see PROJECT DELIVERABLES details below)

Homework must be typed UNLESS your handwriting is extremely readable!  
Late homework (received in e-mail by 7am the following morning) will be penalized one partial grade (e.g. from A- to B+). Late homework received within 48 hrs after class ends will be penalized two partial grades (e.g. from A- to B). Homework received up to 5 days late will be penalized one full grade (e.g. from A- to B-). Homework will not be accepted more than 5 days late, or if a solution to the homework has already been posted.

Recommended Reading

**Bowman (The Practical SQL, 4th Ed)**

* Chapter 11, Solving Business Problems

**Models and Languages for Database Design & Theory**

Visual & Textual Languages for ER Modeling

* Sections 7 Subclasses
* Section 8 Comparative Analysis of UML, Chen & Crow Magnum

**Connolly & Begg**

* Section 12.1 Enhanced ER Modeling
* [OPTIONAL] Chapter 4 Relational Algebra & Calculus

**Oracle Database 10g**

* Chapter 14 Complex Possibilities

Project Continuation

This week, you will be adding superclasses/subclasses to your conceptual model, modifying your relational schema accordingly, and planning how you will initially populate your database

**PROJECT DELIVERABLES**

🡺 Your team name and/or project name must appear at the beginning of all .sql files which you created, and on each team document.

**E-mail a zip file** of your project folder by 11pm on the night *before* the start of class, with the subject **HW #8 from *teamname.*** Note: Make sure that the paths of all files in the zip file are relative to the project folder root  
  
**🡺 READ CAREFULLY**: Also, at the start of class, hand in **HARDCOPY** of

* Your Project Status Document (including your team report)
* Your latest specifications (which should not include your team report)
* 🡺 DefineAppSystem.sql
* 🡺 Your population plan

ALL PROJECT HOMEWORK MUST BE TYPED. HANDWRITTEN MATERIAL WILL NOT BE ACCEPTED (unless you check with me first) 🡺 PLEASE provide everything you hand in for your project in a single packet, either stapled, or put in a pocket folder

🡺 Please hand in project deliverables separately from other homework

**PROJECT GRADING**

Grading for this week's assignment will be based on the following:

* Project Status Document
* Revised Specifications
* Revised Table Creation Code
* Populate Plan

Each individual's grade for the project will be a combination of (a) the grade for the project as a whole, and (b) the contribution that you made to the project (in terms of quality, quantity, and leadership). Individual project grades will be determined at the end of the term.

It is your individual responsibility to ensure that you have an opportunity to participate in and review each aspect of the project, since that will both affect your project grade, as well as your ability to do well on exams. If there are any problems or issues, let me know as soon as possible.

It is the team leader's responsibility to make sure that (a) the team succeeds, and (b) you do your best to see that each person on the team has the opportunity to participate in and/or review each aspect of the project. If you have any concerns about your team's success, or the success of any member of your team, it is your responsibility to let me know as soon as possible. Either the team or I can change the team leader if necessary.

**PROJECT TASKS**

**1)** (if your team has > 1 person) A team participation report (written by the team leader), listing the team name for your group, the name of your project (if different from the team name), the name of the team leader, and for each member of the team

* a very brief description of the main things that member focused on this week
* the percentage of the overall work that member did this week

**2)** Revise the specifications for your application as necessary. **Highlight any changes.**

Your specifications must include the following revisions:

* Your ER model must be revised to 🡺 **include at least one subclass**. You will **get more credit** if both the superclass and the subclass have relationships with other entity classes. Include design notes for any interesting or complex design decisions, including why you did or didn't use subclass discriminators**.**
* Your relational schema must be revised to represent the changes in your ER model. 🡺 **NOTE: You may NOT use a superclass-only representation for *all* of your superclass/subclass relationships.** At least one of them must use a superclass/subclass or a subclass-only representation.
* Revise your listof User Operations so that **at least one of your operations accesses information in a table corresponding to a superclass, and at least one (possibly the same one) will accesses information in a table corresponding to a subclass**. Remember that you need to discuss with me any operation which you believe will require user interaction beyond filling in the initial parameters.

**3)** Update the code to create the tables and views for your application (i.e. using CREATE TABLE and CREATE OR REPLACE VIEW) to correspond to the changes in your specifications.

**4)** Test that your code to create your tables still works. Do the following

* Using SQL\*Plus, connect to the database as the new user you created for your application (e.g. ZooDB), with password OK (or just connect as SCOTT or your user if you were unable to create a new database user).
* Run the application setup script you created (e.g. @/zooset)
* Run (in SQL\*Plus)  
    
   @&ac

This runs create.sql (which you can look at), which builds the core subsystem code, and then runs the code in AppSubsystems\DefineAppSystem.sql, which should create your tables.

Running @&ac automatically creates the spool file Spool\create.LST. Please do not change or fake its content. 🡺 It must be included in your zipped deliverable.

Very carefully look through the script file to see if Oracle reported any errors. If there are errors, fix them, then keep executing @&ac until there are no errors. Read Hints for Debugging with SQL\*Plus and PL/SQL (in courseinfo) for help and advice.

**5)** In the next assignment, you will populate your database with actual data -- enough to do a decent job of testing your application without going overboard.

🡺 In preparation, this week you will write 🡺 and place in Doc\PopPlan.doc, a thoughtful plan (about a page) that specifies how you will populate the database, including

* + how much data there will be for the various tables
  + the kind of data you will use
  + any interesting relationships among the data
  + the order in which you will insert the data

You will have some choices about how to generate the data that you need to insert:

* You can insert each tuple explicitly (e.g. via INSERT statements). That's how the Employee Database Application works, as you can see in \_EmpApp\Populate\populate.sql.
* You can read the data from a file, which will require uses of a PL/SQL package.
* You can automatically generate some of the data using PL/SQL code. For example, you might populate a table using a FOR loop. This could lead to boring names such as Employee1, Employee2, etc., but can be a fast way to get a large number of tuples created

Have fun. Don't go overboard. It's often possible to do a reasonable job of checking that your operations work with *very little data* (but not too little). Remember it's up to you to make sure that populating the database does not become a time sink!

Begin working out the details of the actual data you will use to populate the database. That will save time next week, when you will have more to do.

**6)** Write a **Project Status Document** (🡺 in Doc/status-8.doc), including the elements below (please follow the format of the sample report provided in courseinfo)

Problems

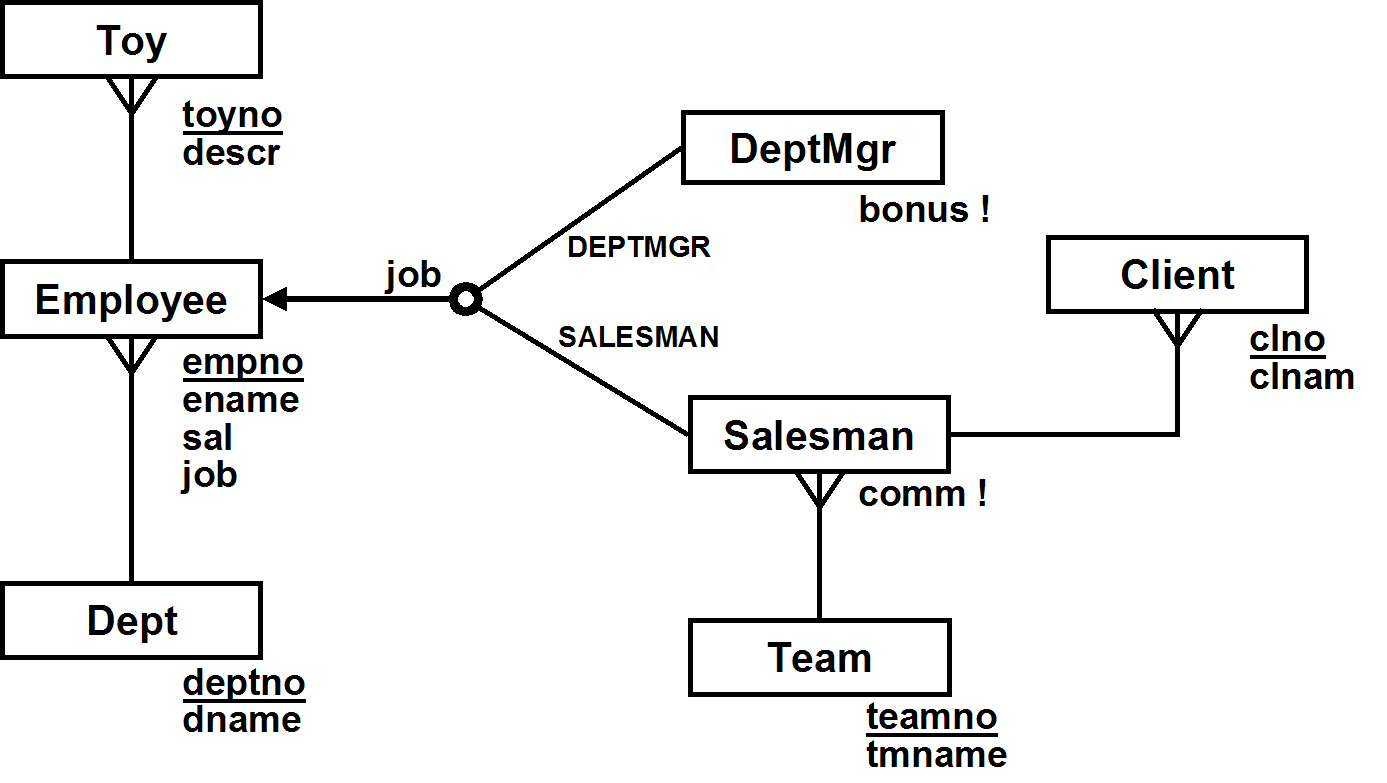
IMPORTANT: You MAY do these problems on your own or with a partner.

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.

**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.

**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.

**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.

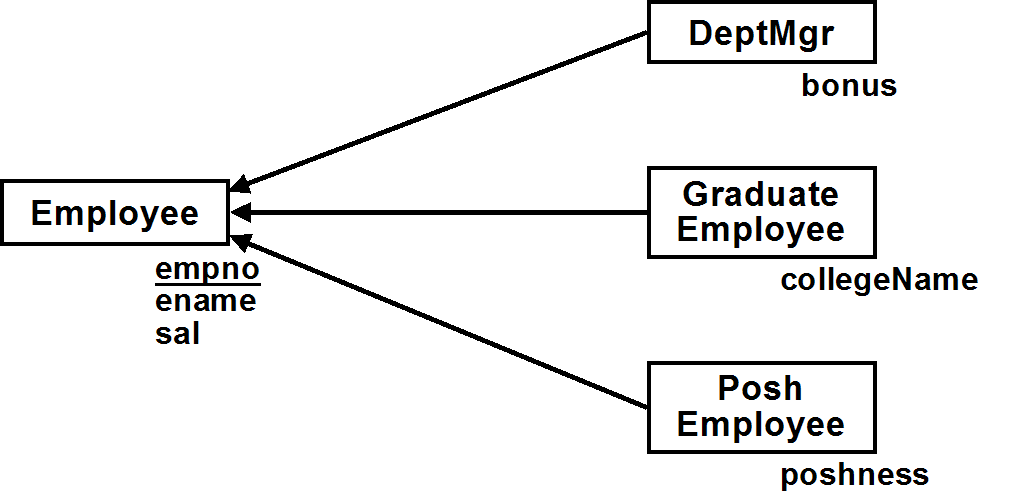
🡺 You MUST use the following entity classes: Lesson, Game, Instructor and Member. You will need additional entity classes as well. In particular, consider classes which are superclasses of the ones above.

**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.