# **CS579 Fall 2009**

# Class #3 Homework

**Submitter: Charlie Kim**

**1. [35%]** **SQL Grouping**

Using the Emps and Depts table you created in exercise #1, write and **test** SQL that will solve the problems below.

🡺 **IMPORTANT:** Make sure that the actual code that you write is independent of the actual data in the table. That, your code should do the right thing even if there were a different set of employees in the table

🡺 For full credit, write any required joins using NATURAL JOIN expressions

a) Display the count of the number of employees at each location loc

**SELECT loc, count(\*) AS num\_emps**

**FROM emps NATURAL JOIN depts.**

**GROUP BY loc ;**

b) Display the count of the number of distinct salary values that occur in each department

**SELECT dname, COUNT(DISTINCT sal) AS uniq\_sals**

**FROM emps NATURAL JOIN depts.**

**GROUP BY dname ;**

c) For each unique combination of location and jobs, list the minimum and total salary of those who have that job at that location.

**SELECT loc, job, MIN(sal) as min\_sal, SUM(sal) as sum\_sal**

**FROM emps NATURAL JOIN depts.**

**GROUP BY loc, job ;**

d) List the jobs in each department where only one person in the department has the job. (You can assume that every employee has a job)

**SELECT dname, job**

**FROM emps NATURAL JOIN depts.**

**GROUP BY dname, job**

**HAVING COUNT(job) = 1 ;**

e) For each location loc where there is only a single department located, display the count of the number of different jobs at that location

**SELECT loc, COUNT(DISTINCT job)**

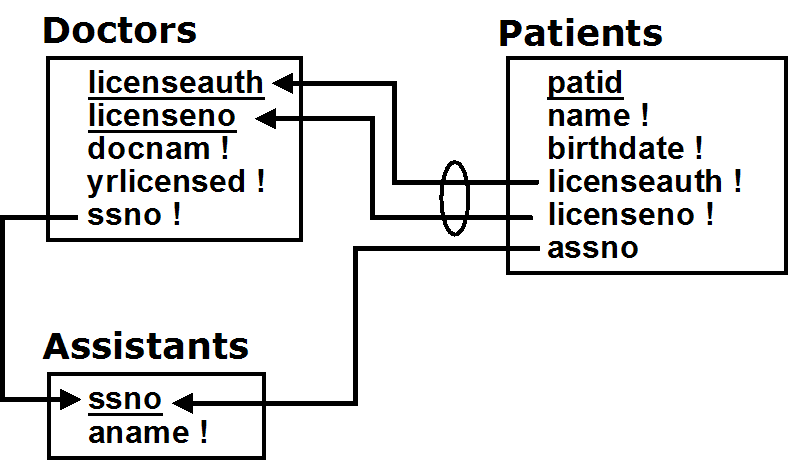
**FROM emps NATURAL JOIN depts.**

**GROUP BY loc**

**HAVING COUNT(DISTINCT deptno) = 1 ;**

**2. [30%] Relational Implementation & Reverse Engineering**

Look at the following relational schema diagram, which describes a situation in which doctors treat patients and are assisted by assistants who medicate patients, and do parts (a), (b) and (c).



**a)** Write and **test** the SQL to create the three tables which correspond to this relational schema. Make sure that it includes all the appropriate

* primary key constraints   
  (NOTE: There are primary key constraints, but no explicit UNIQUE constraints)
* not null constraints
* and foreign key constraints (using **references**)

which are shown in the relational schema above.

🡺 **To get full credit, you must** **include the output from the spool file, showing successful creation of the tables.**

**🡺 You must create a table before you refer to it or else you will get errors!**

For extra credit, provide names for the constraints.

It is very useful to provide names for constraints, especially foreign key and check constraints. If there's an error, this name is reported in error messages. (Primary key and null constraints can be named as well, but they're less likely to be violated).

**SCOTT>**

**SCOTT> -- hw3 2a**

**SCOTT>**

**SCOTT> drop table patients;**

**drop table patients**

**\***

**ERROR at line 1:**

**ORA-00942: table or view does not exist**

**SCOTT> drop table doctors;**

**drop table doctors**

**\***

**ERROR at line 1:**

**ORA-00942: table or view does not exist**

**SCOTT> drop table assistants;**

**drop table assistants**

**\***

**ERROR at line 1:**

**ORA-00942: table or view does not exist**

**SCOTT>**

**SCOTT> create table assistants (**

**2 ssno number(10) primary key,**

**3 aname varchar2(32) not null**

**4 );**

**Table created.**

**SCOTT>**

**SCOTT>**

**SCOTT> create table doctors (**

**2 licenseauth number(10),**

**3 licenseno number(10),**

**4 docnam varchar2(32) not null,**

**5 yrlicensed number(2) not null,**

**6 ssno number(10) references assistants,**

**7 primary key ( licenseauth, licenseno )**

**8 );**

**Table created.**

**SCOTT>**

**SCOTT> create table patients (**

**2 patid number(10) primary key,**

**3 name varchar2(32) not null,**

**4 birthdate date not null,**

**5 licenseauth number(10) not null,**

**6 licenseno number(10) not null,**

**7 assno number(10) references assistants(ssno),**

**8 foreign key ( licenseauth, licenseno ) references doctors**

**9 );**

**Table created.**

**SCOTT>**

**SCOTT>**

**SCOTT> spool off**

**b)** Write and **test** the SQL to insert three tuples   
 (and **include the spool file** for that as well)

1. a patient who sees a doctor and is medicated by an assistant
2. that doctor
3. the assistant to that doctor who medicates the patient

NOTE: Because of the foreign key constraints you (should have) specified when you defined the tables, you must be careful to write the inserts in an order that will actually allow them to all be done without raising an error! Also, do NOT use a SELECT as part of any INSERT.

**SCOTT> -- hw3 2c**

**SCOTT> insert into assistants values(6001,'Assist A');**

**1 row created.**

**Commit complete.**

**SCOTT> insert into doctors values(3030,8703001,'Dr. Nobody', 15, 6001);**

**1 row created.**

**Commit complete.**

**SCOTT> insert into patients values(10003,'Nice Pati','3-Sep-1999',3030,8703001,6001);**

**1 row created.**

**Commit complete.**

**SCOTT>**

**SCOTT> spool off**

**c)** Draw a (Easy) Crow Magnum or UML ER Diagram corresponding to the relational schema above, based on carefully reverse engineering it. You must

* Show all attributes, making sure you indicate primary keys, and attributes which are required to have values.
* Be careful NOT to include entity attributes!
* Show all relationship characterizations (make sure the names are appropriate)
* Show mandatory [but NOT optional] participation constraints.

Patient

sees

< helps

patid

name !

birthdate !

Doctor

Assistant

^

assists

ssno

aname !

licenseauth

licenseno

docnam !

yrlicensed !

**3. [35%] Requirements & Conceptual State Constraints**

Below, you will find requirements for an Employee Database Application, and the ER model designed to implement those requirements. 🡺 I strongly recommend that you read through the solution to the conditions and constraints problem in HW #2 before doing this problem!

This problem asks you to extract all the **conceptual state constraints** from the requirements ***except*** those already visually represented by the ER diagram. You must separate out entity instance constraints, entity class constraints, relationship constraints, and general conceptual constraints.

Only include those conceptual constraints which appear in, or are implied by the requirements. Remember that

* conceptual constraints are written in plain non-technical English
* conceptual constraints involve the conceptual model (as show above). They refer to entity classes & relationships, not to tables!
* a state constraint specifies an invariant property of the database state (*something that you would like to always be true*). It must be **possible** to check a state constraint by looking at the database *at an arbitrary time*, so there must be attributes or relationships in the conceptual model which can be tested ***at any time*** to determine if the state constraint is true or not.
* a state constraint *constrains*. That is, it limits the states of the database which are allowable. A statement which does not constrain anything can't be a constraint.

Do NOT include

* pre-conditions: Business rules which can ONLY be checked prior to specific operations, determining whether or not the operation is allowed
* post-conditions: Business rules which ONLY describe how the state has changed after doing an operation, but which are not always true at an arbitrary point in time
* state constraints which are already visually represented by the ER diagram (e.g. mandatory participation constraints)

NOTE: To provide additional help, following this problem you will find the filled-in solutions for an Online Job Board Application

**Employee Database Application Requirements**

An Employee Database Application keeps track of all employees of a small company, including their name, job and salary. The company is divided into departments (with a name and location), and every employee works for a single department.

Only employees can successfully login to the system, however, even without logging in, a user can get the names and locations of all the company departments.

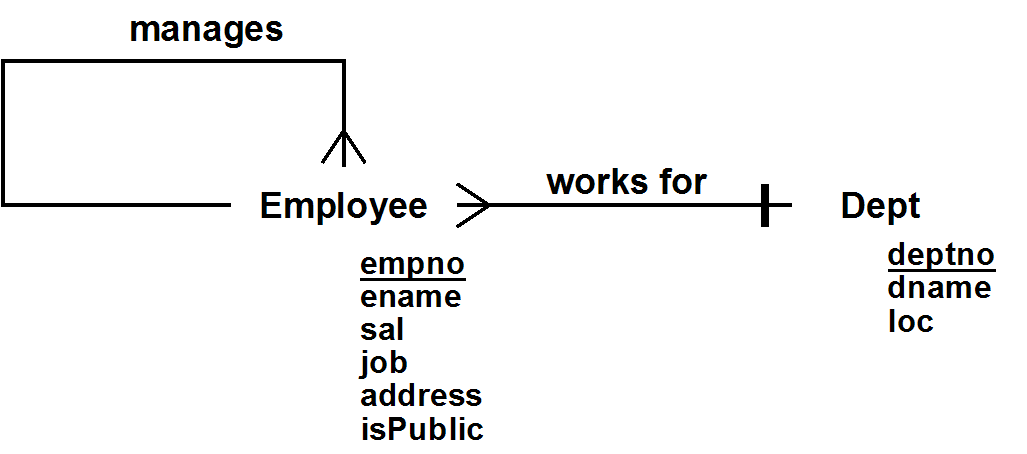
An employee can optionally provide and change their own address, and can see the address of any other employee who has provided an address and made it public, given that employee's name. An employee can see a list of the names and jobs of all the employees in their own department, and can find out the department and location of another employee given their name.

Every employee (except for the President – the employee whose job is 'PRESIDENT') has a manager, who directly manages them. An employee does not need to be in the same department as their direct manager. However, every employee works at the same location as their direct manager, unless they are directly managed by the President. When a manager is terminated, all employees who directly reported to them instead will instead report to their old manager’s manager. Managers can get a list of all the employees that they directly manage, or all employees beneath them in the management hierarchy, along with the job, salary and manager of each.

Every department has at most one department manager (the employee in a department whose job is 'DEPTMGR'), and must have a department manager if it has a clerk. Any employee can find out the managers of the various departments. Every department manager may also directly manage employees, and all department managers are directly managed by the president. A department manager can see the job, salary and manager of all other employees in their department, and can change the jobs and salaries of all the other employees in their department, and terminate employees in their department. A department manager can hire a new employee for their department, only if their initial direct manager is also in the same department. A department manager can change the direct manager of an employee in their department only if both the old and the new manager are in the same department. When a department loses its manager, the employee in the department with the highest salary (or one of them if there is more than one) becomes the manager, if one is needed.

The president can see and change the jobs, salaries, managers, and departments of all other employees, and can hire and terminate any other employee. The President can also create and destroy departments and move them to a new location. When destroying a department, the President can choose to either terminate all the employees in the department, or move them all together to a different department. However, no operation can affect the president.

**ER Model:**



🡺 HERE ARE THE QUESTIONS FOR THIS HOMEWORK PROBLEM:

**a) Entity Instance Constraints**

a1) List all entity instance constraints on **Employee**. These are conceptual state constraints which constrain the value of a single Employee at a time, independent of any relationship

**Every employee must have a name.**

a2) List all entity instance constraints on **Dept**. These are conceptual state constraints which constrain the value of a single Dept at a time, independent of any relationship

**Every department must have a name.**

**b) Entity Class Constraints**

b1) List all entity class constraints on **Employee**. These are conceptual state constraints which only involve the Employee class, independent of any relationship, but which require consideration of more than one Employee at a time

**NONE**

b2) List all entity class constraints on **Dept**. These are conceptual state constraints which only involve the Dept class, independent of any relationship, but which require consideration of more than one Dept at a time

**NONE**

**c) Relationship Constraints**

c1) List (and name) all relationship constraints on the **Manages** relationship:

Note: These are conceptual state constraints which involve the Manages relationship (see the list of named relationships above) and possibly the class(es) it relates (in this case only the Employee class; no other classes or relationships).

**PresidentialManage**

**All department managers are directly managed by the president.**

**MustHaveManager**

**Every employee must have a manager except the president.**

c2) List (and name) all relationship constraints on the **WorksFor** relationship:

Note: These are conceptual state constraints which involve the WorksFor relationship (see the list of named relationships above) and possibly the class(es) it relates (the Employee & Dept classes, but no other relationships).

**MaxOneDeptMgrPerDept**

**Every department has at most one department manager**

**DeptMgrIfCleark**

**Every department must have a department manager if it has a clerk.**

**d) General Conceptual State Constraints.** List all general conceptual state constraints. These are conceptual state constraints which involve two or more relationships (or two or more unrelated classes)

**CoLocDirectMgr**

**Employees must work at the same location as their direct manager.**

**Online Job Board Application Requirements**

An Online Job Board Application allows job candidates and company representatives to post information and find possible matches.

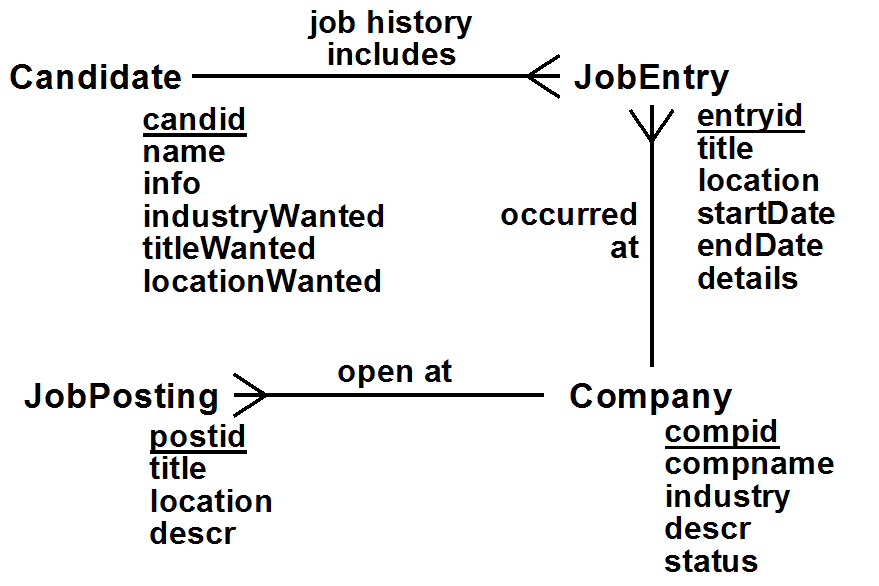
After a company representative registers their company, they can provide and later update information about it, including the industry it is in (chosen from a standard list) and a description. They can also post, update and remove job openings, each one describing the title and location of the job and an optional description. Two job openings for the same company must not have the same title and location.

A company representative can deactivate their company (and later, after arranging payment, reactivate it or make it’s status premium). When a company is deactivated, all of that company’s job postings are removed. The only company-representative-specific operations that can be done by a company representative for an inactivate company are to update the company information and reactivate the company. Excluding inactive companies, only 30% of the companies within an industry can have PREMIUM status. Only companies with PREMIUM status can have more than 3 jobs posted.

After a job candidate registers, they can provide and later update their name and other personal information, as well as the industry, title and location of the job they are seeking. They can also provide information about each of the jobs they have had, including the start and end dates (unless they are still at that job), the company, and optionally their job title, location, and other details. If the company they worked is not already in the database, they can provide information about it. A candidate may have more than one job entry for the same company, but if so, their dates of employment must not overlap. If a candidates removes themselves from the system, their job history is removed as well.

A company representative for an active company can browse and search through employees and see their information. A job candidate can find job openings that match their desires. Either can both browse and search through companies and lists of open jobs.

**ER Diagram**



**Relationships**

JobHistory: Candidate *job history includes* (\*) JobEntry

OccurredAt: (\*) JobEntry *occurred at* Company

OpenPostings: (\*) JobPosting *open at* Company

**a) Entity Instance Constraints**

a1) Candidate

* Every candidate must have a name
* IndustryWanted must be one of the approved industry values

a2) Company

* Industry must be one of the approved industry values
* Status is either PREMIUIM, ACTIVE or INACTIVE

a3) JobEntry

* Either no endDate is provided, or it must be greater than the startDate
* StartDate must be provided

a4) JobPosting

* Both the title and the location must be provided

**b) Entity Class Constraints**

b1) Candidate: None

b2) Company:

🡺 PremiumLimits  
Excluding inactive companies, only 30% of the companies within an industry can have PREMIUM status

b3) JobEntry: None

b4) JobPosting: None

**c) Relationship Constraints**

c1) JobHistory Constraints: None

c2) OccurredAt Constraints: None

c3) OpenPostings Constraints:

🡺 UniqueOpenings  
Two job openings for the same company must not have the same title and location

🡺 PremiumPostings  
Only companies with premium status can have more than 3 jobs posted

**d) General Conceptual Constraints**

🡺 NoOverlappingDates  
If a candidate has two job entries for the same company, the dates must not overlap(involves both the JobHistory & OccuredAt relationship)