CS 348 Introduction to Database Management
Assignment 2

Due: 30 October 2012 9:00AM
Returned: 8 November 2012
Appeal deadline: One week after return
Lead TA: Jiewen Wu

Submission Instructions: By the indicated deadline you are to use the submit command to submit a file containing SQL queries that implement each of the requests for information given in Question 1. You are also to submit written or typed answers to Question 2 in the assignment boxes. For the online submission, put all your queries (in the order specified in this handout) in one file named a2.sql. We should be able to run the queries using the command `db2 -f a2.sql`. Assume a database connection already exists (i.e., you do not need `connect` or `disconnect` statements in your submitted file). To submit your assignment use `submit cs348 a2 .` (notice the dot "." at the end indicating that you submit the entire directory).

Please submit the answers to second and third questions by placing hard copy solutions in the assignment boxes.

Question 1.

For the first question, you must use your Unix accounts and DB2 to compose and evaluate a number of SQL queries for a database that records information about courses. The schema for the database is illustrated in Figurereffig:q1 that depicts a relational database schema that includes an indication of primary and foreign key constraints in the manner discussed in class. Sample commands for defining the base tables for this schema can be downloaded from the course web site (go to the Assignments tab). Note that the schema stores information about both ongoing and past classes for a course. Also note that no marks are recorded for any enrollment of an ongoing class, and that, for a past class, a mark is recorded for each of its enrollments. Finally, you may assume that each class has at least one enrollment.

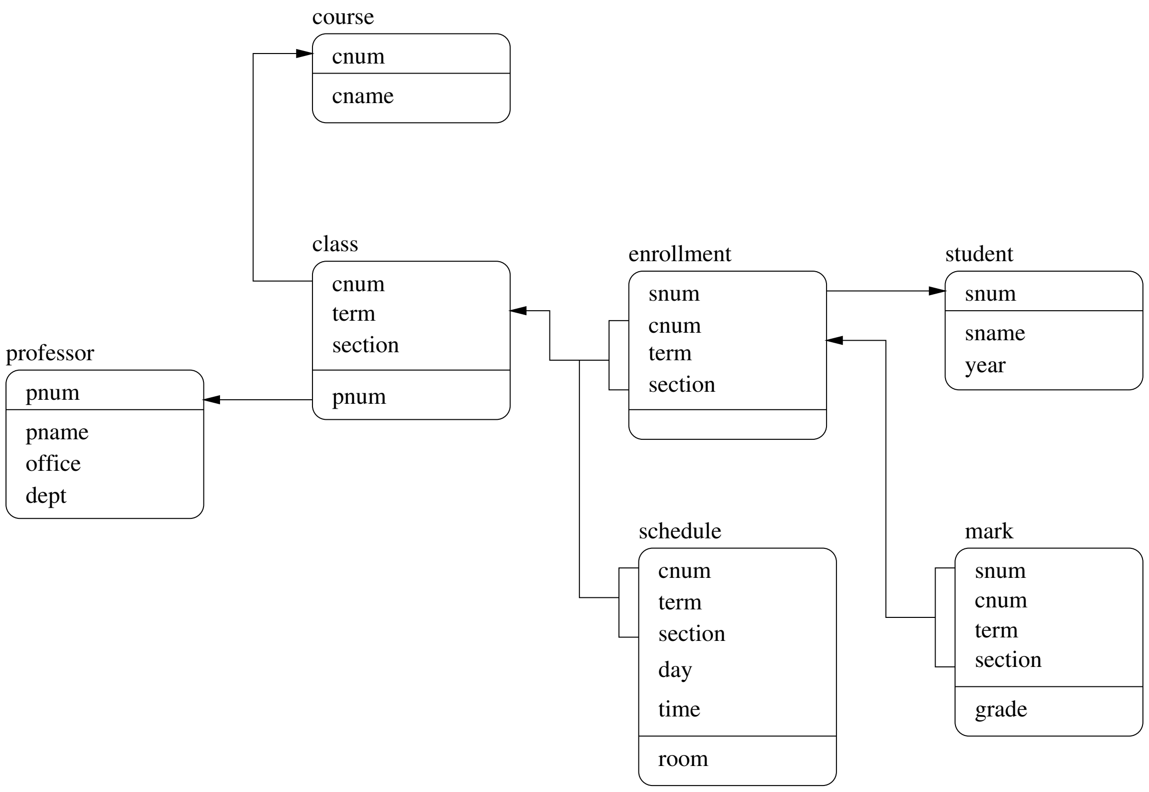

Figure 1: Schema for Question 1

(a) The number, name and department of each instructor who has taught in the past on Mondays.

Solution:

```
select distinct p.pnum, p.pname, p.dept
from professor as p,
        class as c,
        schedule as s
where exists
        (select *
         from mark as m, enrollment as e
         where c.cnum = e.cnum
         and c.term = e.term
         and c.section = e.section
         and e.cnum = m.cnum
         and e.term = m.term
         and e.section = m.section)
and c.cnum = s.cnum
and     c.term = s.term
and     c.section = s.section
and     s.day = 'Monday'
and     c.pnum = p.pnum
```

(b) The number of instructors who are currently teaching CS348.

Solution:

```
select count(p.pnum) as Total
from enrollment as e,
        professor as p,
        class as c,
        student as s
where c.cnum = 'CS348'
and c.cnum = e.cnum
and c.term = e.term
and c.section = e.section
and s.snum = e.snum
and p.pnum = c.pnum
and not exists
    (select * from mark
     where mark.cnum = e.cnum
     and     mark.term = e.term
     and     mark.section = e.section)
```

(c) The course number, course name and grade of each course completed by a student whose student number is 1234.

Solution:

```
select c.cnum, c.cname, m.grade
from course as c, class as cl,
     mark as m, enrollment as e
where e.snum = 1234
and e.snum = m.snum
and cl.cnum = e.cnum
and cl.term = e.term
and cl.section = e.section
and e.cnum = m.cnum
and e.term = m.term
and e.section = m.section
and cl.cnum = c.cnum
```

(d) The number, name and year of each student who is not in his or her first year, who has a final grade of at most 70 in every course that she/he has completed and who was not taught by a professor in the philosophy department (including current courses). Note: use department name "Philosophy".

Solution:

```
select snum, sname, year
from student as s
where year <> 1
and exists
    (select *
     from mark as m
     where m.snum = s.snum)
and not exists
    (select *
     from mark as m
     where m.snum = s.snum
     and m.grade > 70)
and not exists
    (select *
     from professor as p, class as cl, enrollment as e
     where p.dept = 'Philosophy'
     and p.pnum = cl.pnum
     and cl.cnum = e.cnum
     and cl.term = e.term
     and cl.section = e.section
     and e.snum = s.snum)
```

(e) The number, name and department of each professor who has no current teaching commitments on either Mondays or Fridays, sorted by department and then by the professor's name. Note: use full day names, e.g., "Monday".

Solution:

```
select p.pnum, p.pname, p.dept
from professor p
where not exists
    (select *
     from class c, schedule s
    where c.cnum = s.cnum
    and c.term = s.term
    and c.section = s.section
    and c.pnum = p.pnum
    and s.day in ('Monday', 'Friday')
    and not exists
        (select *
         from mark m, enrollment as e
         where e.cnum = c.cnum
         and e.term = c.term
         and e.section = c.section
         and m.cnum = e.cnum
         and m.term = e.term
         and m.section = e.section)
    )
order by p.dept, p.pname
```

(f) The course number, term and section number of each class taught in the past by some professor, together with the number and name of that professor.

Solution:

```
select distinct c.cnum, c.term, c.section, p.pnum, p.pname
from professor p, class c
where p.pnum = c.pnum
and exists
    (select *
     from mark as m, enrollment as e
     where e.cnum = c.cnum
     and e.term = c.term
     and e.section = c.section
     and m.cnum = e.cnum
     and m.term = e.term
     and m.section = e.section)
```

(g) The course numbers and total enrollment of courses with total enrollment counts among the three lowest. (Note that one possible result could be: ['CS448', 120], ['CS446', 120], ['CS341', 105], ['CS246', 110]. Also note that all classes, past and ongoing, need to be considered.)

Solution:

```
select e.cnum, count(*) as numOfEnrollment
from enrollment e
group by e.cnum
having count(*) in (
    select a.c
    from (select distinct count(*) as c
        from enrollment e group by e.cnum) a
    where
        (select distinct count(*)
         from (select distinct count(*) as c
             from enrollment e group by e.cnum) b
         where a.c > b.c) < 3)
```

(h) The number of different third year students in each section of each course taught by each professor in the current term. The result should include the professor number, professor name, course number and section, and should also be sorted first by the name of the professor, then by the professor number, third by the course number, and finally by section. (Note that a section is identified by a term and a section number. Also assume that sorting by section means sorting by term and then by section number. The result will therefore have a total of six columns.)

Solution:

```
select p.pname, p.pnum, cl.cnum, cl.term, cl.section,
      count(*) as numOfStudent
from student as s, professor as p,
        class as cl, enrollment as e
where s.year = 3 and e.snum=s.snum
and e.cnum=cl.cnum and e.term=cl.term
and e.section=cl.section and p.pnum=cl.pnum
and not exists
   (select *
    from mark m, enrollment as e2
    where cl.cnum=e2.cnum
    and     cl.term=e2.term
    and     cl.section = e2.section
    and     e2.cnum=m.cnum
    and     e2.term=m.term
    and     e2.section = m.section
   )
group by p.pname, p.pnum, cl.cnum, cl.term, cl.section
order by p.pname, p.pnum, cl.cnum, cl.term, cl.section
```

(i) The minimum and maximum final grade for each class that was taught in the past by a professor in the statistics department. The result should include the number and name of the professor, and the course number, course name, term and section of the class. Note: use department name “Statistics”.

Solution:

```
select max(grade) as maxGrade, min(grade) as minGrade,
    p.pnum,p.pname,c.cnum,c.cname,cl.term,cl.section
from professor p, course c, class cl, mark m,
    enrollment as e
where cl.cnum=c.cnum
and p.pnum=cl.pnum
and p.dept = 'Statistics'
and cl.cnum = e.cnum
and cl.term = e.term
and cl.section = e.section
and e.cnum = m.cnum
and e.term = m.term
and e.section = m.section
group by p.pnum,p.pname,c.cnum,c.cname,cl.term,
     cl.section
```

Here is an alternative formulation that does not use aggregates:

```
select maxCourseGrade.grade as Max,
   minCourseGrade.grade as Min, P.pnum, P.pname,
   CL.cnum, CR.cname, CL.term, CL.section
from professor P, course CR, class CL,
   (select cnum, term, section, grade
    from  mark,
      select M1.cnum, M1.term, M1.section, M1.grade
      from mark M1, mark M2
      where M2.grade > M1.grade
      and (M1.cnum = M2.cnum
      and M1.term = M2.term
      and M1.section = M2.section)) as maxCourseGrade,
      (select cnum, term, section, grade
       from  mark
       except
       select M1.cnum, M1.term, M1.section, M1.grade
       from mark M1, mark M2
       where M2.grade < M1.grade
       and (M1.cnum = M2.cnum
       and M1.term = M2.term
       and M1.section = M2.section)) as minCourseGrade
where CL.cnum = CR.cnum
and CL.cnum = maxCourseGrade.cnum
and CL.term = maxCourseGrade.term
and  CL.section = maxCourseGrade.section
and CL.cnum = minCourseGrade.cnum
and CL.term = minCourseGrade.term
and  CL.section = minCourseGrade.section
and P.pnum = CL.pnum
and P.dept = 'Statistics'
```

(j) The percentage of departments with professors who have never taught more than one class in the same term.

Solution:

```
select a.c*100/b.c as percentage
from
     (select count (distinct dept) as c
       from professor
       where pnum not in
           (select pnum
            from class c
            where not exists
              (select *
               from mark m, enrollment as e
               where e.cnum = c.cnum
               and e.term = c.term
               and e.section = c.section
               and m.cnum = e.cnum
               and m.term = e.term
               and m.section = e.section)
            group by term, pnum
            having count(*)<=1) ) as a,
     (select count (distinct dept) as c
         from professor) as b
```

Question 2.

Write queries in the relational algebra for each of the above specifications for which this is possible. (Hint: there are five of them.)

Solution: The algebraic equivalences exist for (a), (c), (d), (f), and (i) if you follow the second version if (i). Here are the solutions

(a) $\Pi_{\text{pnum, pname, dept}}(\text{Professor} \bowtie (\text{Class} \bowtie$
$((\text{Mark} \bowtie \text{Enrollment}) \bowtie \sigma_{\text{day} = \text{`Monday'}}(\text{Schedule}))))$

(c) $\Pi_{\text{cnum,cname,grade}}(\text{Course} \bowtie (\text{Class} \bowtie$
$((\text{Mark} \bowtie \text{Enrollment}) \bowtie \sigma_{\text{snum}=1234}(\text{Student})))$

(d) Answer computed in three lines to make it easier to see (Q is the final result):

First, we find all students who were taught in the past or currently by a professor in Philosophy department.
$\text{Q1} = \Pi_{\text{snum, sname, year}}(\text{Student} \bowtie \text{Enrollment} \bowtie \text{Class} \bowtie \sigma_{\text{dept=Philosophy}}(\text{Professor}))$

Next, we find all students that have at least one grade over 70.
$\text{Q2} = \Pi_{\text{snum,sname,year}}(\text{Student} \bowtie (\text{Enrolled} \bowtie \sigma_{\text{grade}>70}(\text{Mark})))$

Now, in the final answer we want all students who are not in their first year, and who are **not** in the results of Q1 nor Q2.
$\text{Q} = \sigma_{\text{year}>1}(\text{Student}) - (\text{Q1} \cup \text{Q2})$

(f) $\Pi_{\text{cnum,term,section,pnum,pname}}(\text{Professor} \bowtie \text{Class} \bowtie (\text{Enrollment} \bowtie \text{Mark}))$

(i) If you follow the second version of the SQL query, then you can see how to write in relational algebra. I will write it in multiple steps so that you can see it better. In the following, $Q2$ computes the class with the minimum grade, and $Q3$ computes the class with the maximum grade. Q is the final result

The first step is to compute all classes (with student marks) that were taught by a professor in Statistics department.

$$Q1 = \sigma_{\text{dept='Statistics'}}(\text{Professor}) \bowtie (\text{Class} \bowtie (\text{Enrollment} \bowtie \text{Mark}))$$

The following two queries find minimum and maximum grade, respectively, for each class found by Q1. If you look closer at the sub-queries wich start with the projection (i.e. the stuff we want to subtract from Q1) - the results of these sub-queries is a set of grades for each class, such that each grade is greater (less for Q3) than at least one other grade for that class. If we subtract that from Q1 - we will be left with set of grades for each class, such that each grade is not greater (not less for Q3) than any grade for that class - i.e. we found the set of minimum/maximum grades for each class.

$$\begin{aligned} Q2 = \;& Q1 - (\Pi_{\text{pnum1,pname1,office1,dept1,cnum1,term1,section1,snum1,grade1}} \\ & (\sigma_{\text{grade1>grade2}}(\rho_1(F1, Q1) \bowtie_{\text{cnum1=cnum2} \wedge \text{term1=term2} \wedge \text{section1=section2}} \rho_2(F2, Q1)))) \\ Q3 = \;& Q1 - (\Pi_{\text{pnum1,pname1,office1,dept1,cnum1,term1,section1,snum1,grade1}} \\ & (\sigma_{\text{grade1<grade2}}(\rho_1(F1, Q1) \bowtie_{\text{cnum1=cnum2} \wedge \text{term1=term2} \wedge \text{section1=section2}} \rho_2(F2, Q1)))) \end{aligned}$$

Finally, we combine the above results to produce the final answer. Note: it is important to use natural join rather than union to combine results of Q2 and Q3. WIth union, if minimum and maximum grades coincide for a class - that grade will only be reported once, while we want to see it twice.

$$\begin{aligned} Q = \;& \Pi_{pnum,pname,cnum,cname,term,section,min,max} \\ & (Course \bowtie (\rho(grade \to min, Q2) \bowtie \rho(grade \to max, Q3))) \end{aligned}$$

where

$$\begin{aligned} F1 = \; & \{pnum \rightarrow pnum1, pname \rightarrow pname1, office \rightarrow office1, dept \rightarrow dept1, \\ & cnum \rightarrow cnum1, term \rightarrow term1, section \rightarrow section1, snum \rightarrow snum1, \\ & grade \rightarrow grade1\} \\ F2 = \; & \{pnum \rightarrow pnum2, pname \rightarrow pname2, office \rightarrow office2, dept \rightarrow dept2, \\ & cnum \rightarrow cnum2, term \rightarrow term2, section \rightarrow section2, snum \rightarrow snum2, \\ & grade \rightarrow grade2\} \end{aligned}$$

Question 3.

Consider the following schema given in Figure 2 where the underlined attributes are primary keys and the arrows between relations depict the foreign key-primary key relationships. Also consider the following information about relations and give the appropriate DDL statements to create each relation.

Note: A straightforward solution would not allow you to insert tuples into EMPLOYEE and/or DEPARTMENT after creation. So, you need to be careful how you create these tables and modify them after an insertion. If you can figure out how to do this, there is a bonus of 5 points.

- In EMPLOYEE, Fname and Lname can vary in length with a maximum of 15 characters; Minit is a single character; SIN consist of 9 characters; Address is variable length up to 30 characters; Sex is identified as 'M' or 'F'; Salary is a decimal number with 10 digits and 2 digits after the decimal point; Super_Id has the same specification as SIN; Dno is an integer; Bdate is of type date. For each tuple, Fname, Lname, SIN, have to be specified, and each employee has to be assigned to a department.
- In DEPARTMENT, Dname is variable length character string with a maximum length of 15; Mgr_SIN is a fixed length string of 9; Dnumber is an integer; and Mgr_start_date is a date. Dname and Mgr_SIN have to be specified, and every department has to have a manager.
- In DEPT_LOC, Dnumber is the same as what it is in DEPARTMENT, and Dlocation is a variable length character string of at most 15 characters. Both of these have to be specified for any tuple.
- In PROJECT, Pname and Plocation are maximum 15 character strings; Pnumber and Dnum are integers. Pname, Pnumber and Dnum have to be specified for each tuple.
- In WORKS_ON, E_SIN is a 9 character string; Pno is an integer; and Hours is a 3 digit decimal number with one digit after the decimal point. All of these have to be specified for each tuple.
- In DEPENDENT, E_SIN is the same as it is in WORKS_ON; Dependent_name is a character string of up to 15 characters; Sex is identified as 'M' or 'F'; Bdate is a date; and Relationship is a character string of up to 8 characters.

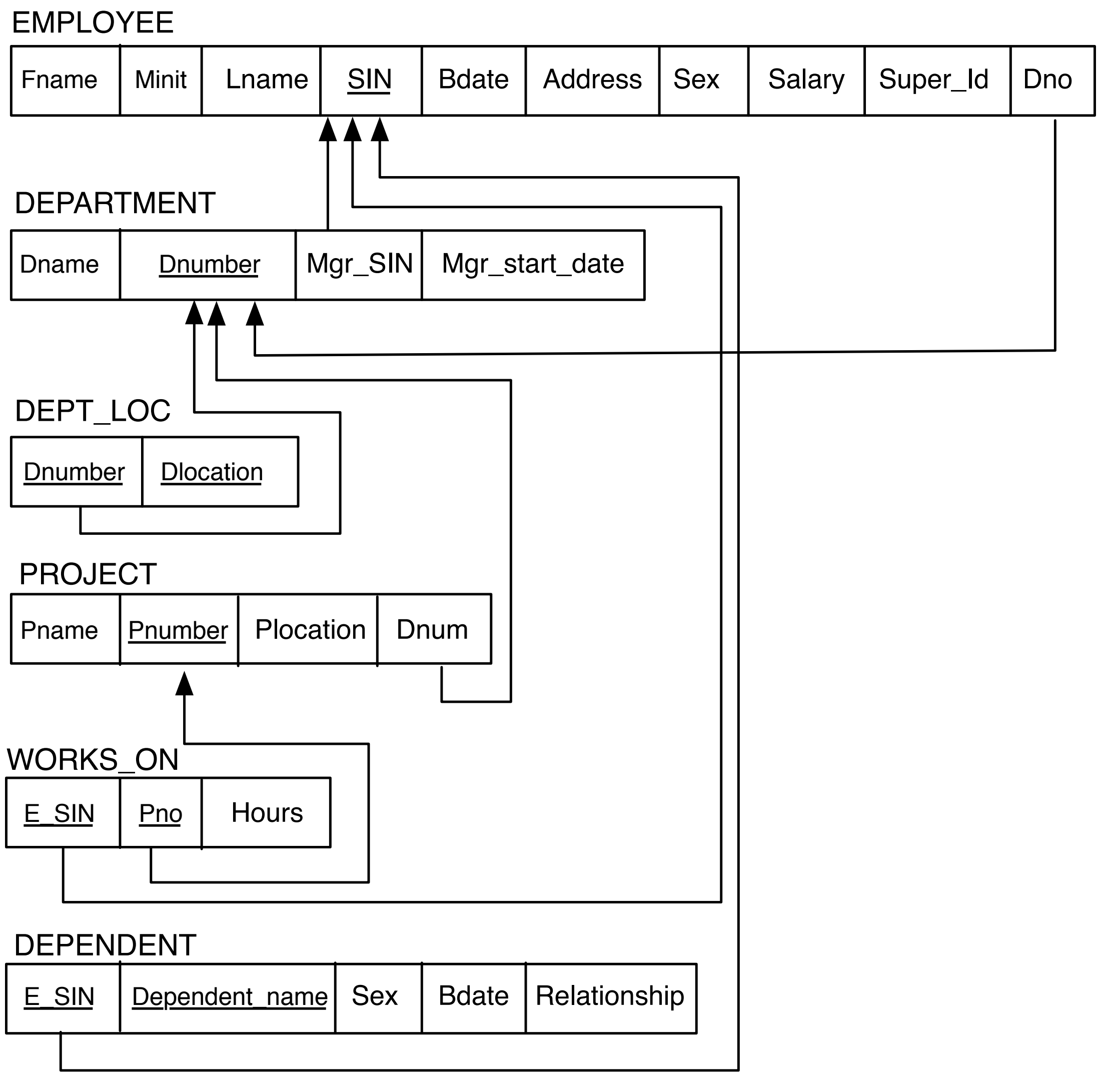

Figure 2: Schema for Question 3

Solution: Here are the required DDL statements:

```
CREATE TABLE EMPLOYEE (
        SIN             char(9) not null,
        Fname           varchar(15) not null,
        Lname           varchar(15) not null,
        Minit           char,
        Bdate           date,
        Address         varchar(30),
        Sex             char,
        Salary          decimal (10, 2),
        Super_Id        char(9),
        Dno             integer not null,
        primary key     (SIN),
)

CREATE TABLE DEPARTMENT (
        Dnumber         integer not null,
        Dname           varchar(15) not null,
        Mgr_SIN         char(9) not null,
        Mgr_start_date  date,
        primary key     (Dnumber),
        foreign key     (Mgr_SIN) references employee(SIN)
)

CREATE TABLE DEPT_LOC (
        Dnumber         integer not null,
        Dlocation       varchar(15) not null,
        primary key     (Dnumber, Dlocation),
        foreign key     (Dnumber) references department (Dnumber)
)

CREATE TABLE PROJECT (
        Pnumber         integer not null,
        Pname           varchar(15) not null,
        Plocation       varchar(15),
        Dnum            integer not null,
        primary key     (Pnumber),
        foreign key     (Dnum) references department (Dnumber)
)

CREATE TABLE WORKS_ON (
        E_SIN           char(9) not null,
        Pno             integer not null,
        Hours           decimal (3, 1) not null,
        primary key      (E_SIN, Pno),
        foreign key      (E_SIN) references employee(SIN),
        foreign key      (Pno) references project(Pnumber)
)

CREATE TABLE DEPENDENT (
        E_SIN            char(9) not null,
        Dependent_name   varchar(15) not null,
        Sex              char,
        Bdate            date,
        Relationship     varchar(8),
        primary key      (E_SIN, Dependent_name),
        foreign key      (E_SIN) references employee(SIN)
)
```

You will notice that we omitted to specify that Dno of the EMPLOYEE relation is a foreign key pointing to the Dnumber of the DEPARTMENT relation. This is because if we don't do that there is circular dependence of insertions and you won't be able to insert tuples into EMPLOYEE or DEPARTMENT (recall that we discussed this in class). So, after you populate DEPARTMENT (i.e., you insert the tuples into it), you need to do a schema change by the following DDL statement:

```
ALTER TABLE EMPLOYEE
ADD FOREIGN KEY
        (Dno) references department(Dnumber)
```