York University **Department of Computer Science and Engineering**	**Test 1 / Fall 2008** **CSE3421**

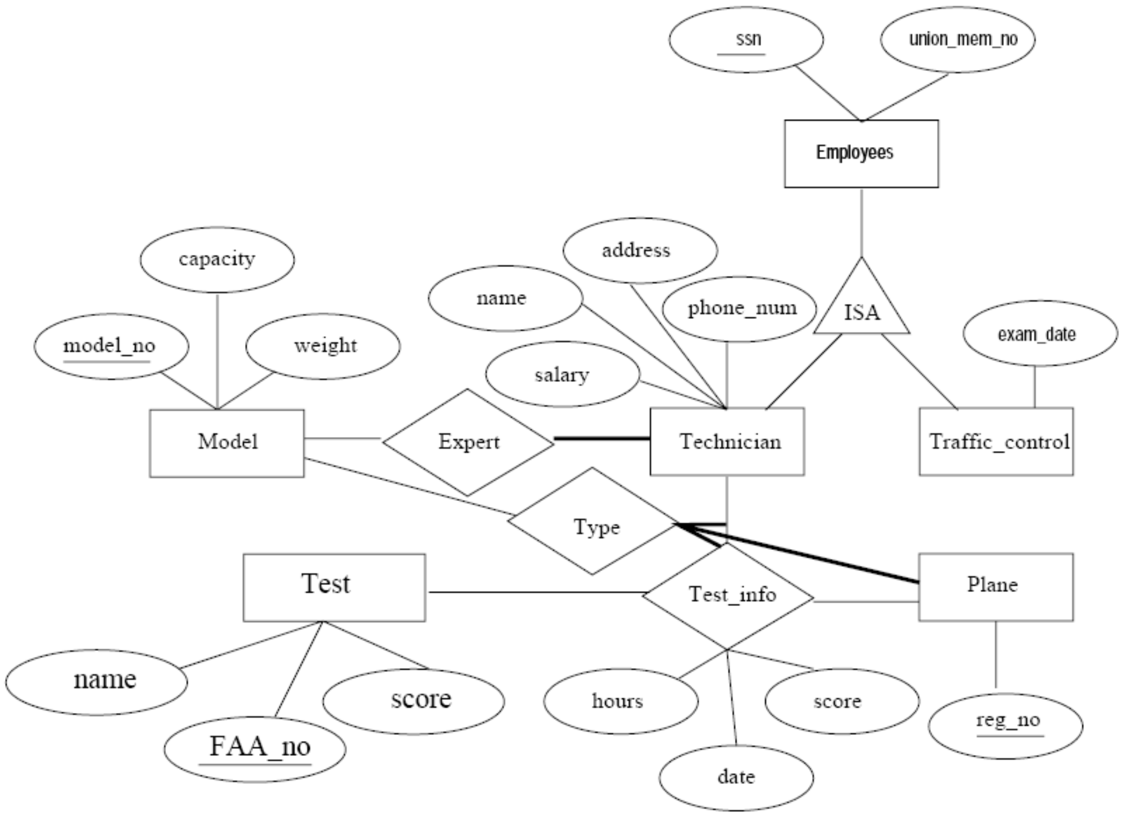
- This is a closed book, **40 minutes** test
- **No questions are allowed during the test. If in doubt, write down your doubts and assumptions and proceed with your answer.**

LAST NAME	SOLUTIONS
FIRST NAME	SOLUTIONS
YORK ID#	SOLUTIONS
CS LOGIN	SOLUTIONS

Exercise 1 [15 points]
The Computer Science Department frequent fliers have been complaining to The Local Airport officials about the poor organization at the airport. As a result, the officials decided that all information related to the airport should be organized using a DBMS, and you have been hired to design the database. Your first task is to organize the information about all the airplanes stationed and maintained at the airport. The relevant information is as follows:

- Every airplane has a registration number, and each airplane is of a specific model.
- The airport accommodates a number of airplane models, and each model is identified by a model number (e.g., DC-10) and has a capacity and a weight.
- A number of technicians work at the airport. You need to store the name, SSN, address, phone number, and salary of each technician.
- Each technician is an expert on one or more plane model(s), and his or her expertise may overlap with that of other technicians. This information about technicians must also be recorded.
- Traffic controllers must have an annual medical examination. For each traffic controller, you must store the date of the most recent exam.
- All airport employees (including technicians) belong to a union. You must store the union membership number of each employee. You can assume that each employee is uniquely identified by a social security number.
- The airport has a number of tests that are used periodically to ensure that airplanes are still airworthy. Each test has a Federal Aviation Administration (FAA) test number, a name, and a maximum possible score.
- The FAA requires the airport to keep track of each time a given airplane is tested by a given technician using a given test. For each testing event, the information needed is the date, the number of hours the technician spent doing the test, and the score the airplane received on the test.

Assume that the following ER diagram has been created from the above description:



The FAA passes a regulation that tests on a plane must be conducted by a technician who is an expert on that model. How would you express this constraint in the given ER diagram? If you cannot express it, explain why not.

Answer:
You cannot note the expert technician constraint the FAA requires in an ER diagram. There is no notation for equivalence in an ER diagram and this is what is needed: the Expert relation must be equivalent to the Type relation.

Exercise 2 [20 points]
For each of the multiple choice questions below circle the letter of the best answer. Do not circle more than 1 answer. Do not add comments or explanations. _Do not guess_: *correct answer = 5, incorrect answer = -5, no answer = 0.*

Given the following schema:

Student(sid:integer, name:string, address : string, age:integer, gpa:real)
sid: student ID; name: student name

Course(cid:string, name:string, department:string, credit:integer)
cid: course id (e.g. cosc3421); name: course name (e.g. "Database Systems"),
department: the department that offers this course; credit: number of credits

Enrolment(sid:integer, cid: string, semester:string, grade:integer)
sid is foreign key referencing sid in student,
cid is foreign key referencing cid in course
semester (e.g. 2002W)

Circle the letter of the single correct answer.

[1] Find the student names who take all the courses:

a. SELECT S.name FROM Student S WHERE NOT EXISTS
((SELECT cid FROM Course C) EXCEPT
(SELECT E.cid FROM Enrolment E WHERE E.sid = S.sid
))

b. SELECT S.name FROM Student S
WHERE EXISTS ((SELECT E.cid FROM Enrolment E WHERE E.sid = S.sid)
EXCEPT (SELECT cid FROM Course C))

c. SELECT S.name FROM Student S WHERE NOT EXISTS
(SELECT cid FROM Course C WHERE EXISTS
(SELECT E.cid FROM Enrolment E WHERE E.sid = S.sid and E.cid = C.cid))

d. SELECT S.name FROM Student S WHERE EXISTS
(SELECT cid FROM Course C WHERE NOT EXISTS
(SELECT E.cid FROM Enrolment E WHERE E.sid = S.sid and E.cid = C.cid))

answer(a)

[2] Find the students who have not taken a 6-credit course:

a. SELECT S.sid FROM Student S WHERE S.sid NOT IN
(SELECT E.sid FROM Enrolment E WHERE E.sid = S.sid
EXCEPT
SELECT E.sid FROM Enrolment E, Course C WHERE E.sid = S.sid and C.credit=6)

b. SELECT S.sid FROM Student S WHERE S.sid NOT IN
(SELECT E.sid FROM Enrolment E WHERE E.cid IN

(SELECT cid FROM Course WHERE credit = 6))

c. SELECT S.sid FROM Student S WHERE S.sid NOT IN
   (SELECT E.sid FROM Enrolment E WHERE E.cid NOT IN
   (SELECT cid FROM Course WHERE credit = 6))

answer (b)

[3] Find the average age of students who are older than 20 for each GPA level:

a. SELECT S.gpa, avg(S.age) FROM Student S WHERE S.age > 20 GROUP BY S.gpa
   HAVING 1 < (SELECT COUNT(*) FROM student S2 where S.sid = S2.sid and S2.age > 20)

b. SELECT S.gpa, avg(S.age) FROM Student S WHERE S.age > 20 GROUP BY S.gpa
   HAVING 1 <
   (SELECT COUNT(*) FROM student S2 where S.gpa = s2.gpa and S2.age > 20)

c. SELECT S.gpa, avg(S.age) FROM Student S WHERE S.age > 20 GROUP BY S.gpa
   HAVING 1 < (SELECT COUNT(*) FROM student S2 where S.sid = S2.sid and S.gpa = S2.gpa and S2.age > 20)

d. SELECT S.gpa, avg(S.age) FROM Student S WHERE S.age > 20 GROUP BY S.gpa
   HAVING 1 <
   (SELECT COUNT(*) FROM student S2 where S.sid = S2.sid and S.gpa = S2.gpa)

answer(b)

[4] Find the students who take at least two courses from the Math department
('MAT'), but none from Chemistry department ('CHM')

```
a.  SELECT   S.sid, S.name
    FROM     Student S
    WHERE    1 < (SELECT COUNT(*) FROM Course C, Enrolment E
                  WHERE E.sid = S.sid and C.cid = E.cid
                                      and C.department = 'MAT'
                  GROUP BY E.sid)
    INTERSECT
    SELECT   S.sid, S.name
    FROM     Student S
    WHERE    NOT EXISTS (SELECT COUNT(*) FROM Course C, Enrolment E
                         WHERE  E.sid = S.sid and C.cid = E.cid
                                          and C.department = 'CHM');
b.  SELECT   S.sid, S.name
    FROM     Student S
    WHERE    1 < (SELECT COUNT(*) FROM Course C, Enrolment E
```

```
                    WHERE E.sid = S.sid and C.cid = E.cid
                                        and C.department = 'MAT'
                    GROUP BY E.sid)
      INTERSECT
      SELECT   S.sid, S.name
      FROM     Student S
      WHERE    NOT EXISTS (SELECT 'X' FROM Course C, Enrolment E
                           WHERE  E.sid = S.sid and C.cid = E.cid
                                                and C.department =
'CHM');
```

```
   c. SELECT   S.sid, S.name
      FROM     Student S
      WHERE    1 < (SELECT COUNT(*) FROM Course C, Enrolment E
                    WHERE E.sid = S.sid and C.cid = E.cid
                                        and C.department = 'MAT'
                    GROUP BY E.sid)
      INTERSECT
      SELECT   '5c', S.sid, S.name
      FROM     Student S
      WHERE    EXISTS (SELECT COUNT(*) FROM Course C, Enrolment E
                       WHERE  E.sid = S.sid and C.cid = E.cid
                                            and C.department <>
'CHM');
```

```
   d. SELECT   S.sid, S.name
      FROM     Student S
      WHERE    1 < (SELECT COUNT(*) FROM Course C, Enrolment E
                    WHERE E.sid = S.sid and C.cid = E.cid
                                        and C.department = 'MAT'
                    GROUP BY E.sid)
      INTERSECT
      SELECT   S.sid, S.name
      FROM     Student S
      WHERE    NOT EXISTS (SELECT 'X' FROM Course C, Enrolment E
                           WHERE  E.sid = S.sid and C.cid = E.cid
                                                and C.department <>
'CHM');
```

Answer is b.

Exercise 3 [30 points]

Consider the relational database with schema as shown below, where the primary keys are underlined.

employee (<u>person-name</u>, street, city)
works (<u>person-name</u>, company-name, salary)
company (<u>company-name</u>, city)
manages (<u>person-name</u>, manager-name)

[a] [10 points] Assume the companies may be located in several cities.

Give an expression in relational algebra to express the following query:
Find all companies located in every city in which "Small Bank Corporation" is located.

Solution:

$$\Pi_{company\text{-}name}\ (company \div (\Pi_{city}\ (\sigma_{company\text{-}name = \text{"Small Bank Corporation"}}\ (company))))$$

[b] [10 points]

Give an expression in relational algebra for each request:
Give all managers in this database a 10 percent salary raise, unless the salary would be greater than $100,000. In such cases, give only a 3 percent raise.

[c] [10 points] Delete all tuples in the works relation for employees of Small Bank Corporation.

Solution

b. The same situation arises here. As before, t_1, holds the tuples to be updated and t_2 holds these tuples in their updated form.

$$t_1 \leftarrow \Pi_{works.person\text{-}name, company\text{-}name, salary}\ (\sigma_{works.person\text{-}name = manager\text{-}name}(works \times manages))$$

$$t_2 \leftarrow \Pi_{works.person\text{-}name, company\text{-}name, salary * 1.03}\ (\sigma_{t_1.salary\ *\ 1.1\ >\ 100000}(t_1))$$

$$t_2 \leftarrow t_2 \cup (\Pi_{works.person\text{-}name, company\text{-}name, salary * 1.1}\ (\sigma_{t_1.salary\ *\ 1.1\ \leq\ 100000}(t_1)))$$

$$works \leftarrow (works - t_1) \cup t_2$$

c. $works \leftarrow works - \sigma_{company-name = \text{"Small Bank Corporation"}}(works)$