Comp115 Spring 2017, HW3 Answer Key

Problem 1 - 25%

Consider the following schema:

Publishers(*pid:* `integer`, *pname:*`string`, *address:* `string`)
Books(*bid:* `integer`, *bname:*`string`, *length:* `integer`)
TuftsBookstore(*pid:* `integer`, *bid:*`integer`, *cost:* `real`)

The key fields are underlined and the domain of each field is listed after the field name. Therefore *bid* is the key for Books, *pid* is the key for Publishers, and *bid* and *pid* together form the key for TuftsBookstore. The TuftsBookstore relationship lists the prices charged for textbooks by Publishers. Write the following queries in relational algebra:

1. Find the *names* of publishers who supply books over 1000 pages long.
   (a) $\pi_{pname}(\pi_{pid}(\pi_{bid}(\sigma_{(length>1000)}Books) \bowtie TuftsBookstore) \bowtie Publishers)$
2. Find the *pids* of publishers who supply books under 200 or over 1000 pages long.
   (a) $\pi_{pid}(\pi_{bid}(\sigma_{(length>1000 \bigvee length<200)}Books) \bowtie TuftsBookstore)$
3. Find the *pids* of publishers who supply books over 1000 pages long and are at 161 College Ave.
   (a) $\pi_{pid}(\sigma_{(address=161CollegeAve)}((\pi_{bid}(\sigma_{(length>1000)}Books) \bowtie TuftsBookstore) \bowtie Publishers))$
4. Find the *pids* of publishers who supply books between 200 and 1000 pages long.
   (a) $\pi_{pid}(\pi_{bid}(\sigma_{(length<1000 \bigwedge length>200)}Books) \bowtie TuftsBookstore)$
5. Find the *pids* of publishers who supply every book.
   (a) $\pi_{pid}(\pi_{bid,pid}(TuftsBookstore)/\pi_{bid}(Books))$
6. Find the *pids* of publishers who supply every book over 1000 pages long.
   (a) $\pi_{pid}(\pi_{bid,pid}(TuftsBookstore)/\pi_{bid}(\sigma_{(length>1000))}Books))$
7. Find the *pids* of publishers who supply every book under 200 or over 1000 pages long.
   (a) $\pi_{pid}(\pi_{bid,pid}(TuftsBookstore)/\pi_{bid}(\sigma_{(length>1000 \bigvee length<200)}Books))$
8. Find the *pids* of publishers who supply every book under 200 pages long or who supply every book over 1000 pages long.
   (a) $(\pi_{(bid,pid)}(TuftsBookstore)/\pi_{bid}(\sigma_{(length>1000)}Books)) \cup (\pi_{(bid,pid)}(TuftsBookstore)/\pi_{bid}(\sigma_{(length<200)}Books))$

9. Find pairs of *pids* such that the publisher with the first *pid* charges more for some book than the publisher with the second *pid*.

   (a) $\pi_{(tb1.pid,tb2.pid)}(\sigma_{(tb1.bid=tb2.bid \bigwedge tb1.cost>tb2.cost)}(TuftsBookstore_{(tb1)} \times TuftsBookstore_{(tb2)}))$

10. Find the *bids* of books that are supplied by at least two different publishers.

    (a) $\pi_{(tb1.bid)}(\sigma_{(tb1.bid=tb2.bid \bigwedge tb1.pid<tb2.pid)}(TuftsBookstore_{(tb1)} \times TuftsBookstore_{(tb2)}))$

11. Find the *bids* of the most expensive books published by Pearsons.

    (a) $((\pi_{pid}(\sigma_{(pname=Pearsons)}(Publishers)) \bowtie (TuftsBookstore))) -$
    $(\pi_{(tb1.bid)}(\sigma_{(t1.cost<t2.cost)}((\pi_{(bid,cost)}((\pi_{pid}(\sigma_{(pname=Pearsons)}(Publishers)) \bowtie$
    $(TuftsBookstore)))_{(t1)} \times (\pi_{(bid,cost)}((\pi_{pid}(\sigma_{(pname=Pearsons)}(Publishers)) \bowtie$
    $(TuftsBookstore)))_{(t2)}))))$

12. Find the *bids* of books supplied by every publisher for less than $200. (If any publisher either does not supply the book or charges more than $200 for it, the book is not selected).

    (a) $\pi_{bid}(\sigma_{(cost<200)}TuftsBookstore) \cap (\pi_{(bid,pid)}(TuftsBookstore)/\pi_{pid}(Publishers))$

Problem 2 - 25%

Please state in lay terms what the following queries compute

1. $\pi_{pname}\,(\pi_{bid}\,(\sigma_{length>200}Books) \bowtie (\sigma_{cost<100}TuftsBookstore) \bowtie Publishers)$

   (a) Answer: Find the names of the publishers who publish books longer than 200 pages and less than 100 dollars that are sold in the book store

2. $\pi_{pname}\,(\pi_{pid}\,((\sigma_{length>200}Books) \bowtie (\sigma_{cost<100}TuftsBookstore)) \bowtie Publishers)$

   (a) Answer: Find the names of the publishers who publish books longer than 200 pages that cost less than 100 dollars

3. $(\pi_{pname}\,((\sigma_{length>200}Books) \bowtie (\sigma_{cost<100}TuftsBookstore) \bowtie Publishers)) \cap$
   $(\pi_{pname}\,((\sigma_{length<300}Books) \bowtie (\sigma_{cost<100}TuftsBookstore) \bowtie Publishers))$

   (a) Answer: Find the names of the publishers that publish books longer than 200 pages that cost less than 100 dollars and publish books shorter than 300 pages that cost less than 100 dollars

4. $(\pi_{pid}\,((\sigma_{length>200}Books) \bowtie (\sigma_{cost<100}TuftsBookstore) \bowtie Publishers)) \cap$
   $(\pi_{pid}\,((\sigma_{length<300}Books) \bowtie (\sigma_{cost<100}TuftsBookstore) \bowtie Publishers))$

   (a) Answer: Find the ids of the publishers who publish books longer than 200 pages that cost less than 100 dollars and publish books that are shorter than 300 pages and cost less than 100 dollars

5. $\pi_{pname}((\pi_{pid,pname}((\sigma_{length>200}Books) \bowtie (\sigma_{cost<100}TuftsBookstore) \bowtie Publishers)) \cap$
$(\pi_{pid,pname}((\sigma_{length<300}Books) \bowtie (\sigma_{cost<100}TuftsBookstore) \bowtie Publishers)))$

   (a) Answer: Find the names of the publishers who publish books longer than 200 pages that cost less than 100 dollars and publish books shorter than 300 pages that cost less than 100 dollars.

Problem 3 - 50%

Consider the following relations:

Student(*snum:* `integer`, *sname:*`string`, *major:*`string`, *level:*`string`, *age:*`integer`)
Class(*name:*`string`, *time:*,`time`, *room:*,`string`, *fid:*`integer`)
Enrolled(*snum:*,`integer`, *cname:*`string`)
Faculty(*fid:*`integer`, *fname:*`string`, *deptid:*`integer`)

The meaning of these relations is straightforward; for example, Enrolled has one record per student-class pair such that the student is enrolled in the class. Write the following queries in SQL. No duplicates should be printed for any answer.

1. Find the names of all students who are enrolled in 0 classes.

   (a) Answer :

```
SELECT s.sname FROM Student s
WHERE s.snum NOT IN
      (SELECT e.snum FROM Enrolled e);
```

2. Find the level of all students who are enrolled in a class that starts before 9:00AM.

   (a) Answer :

```
SELECT s.level
FROM Student s
WHERE s.snum IN
      (SELECT e.snum
       FROM Enrolled e
       WHERE e.cname IN
            (SELECT c.name
             FROM Class c
             WHERE c.time < 9:00))
```

3. Find the course with the most students enrolled that starts before 9:00AM.

   (a) Answer :

```
SELECT c.name
FROM Class c, Enrolled e
```

```
WHERE e.cname = c.name
    AND c.time < 9:00
GROUP BY e.cname
ORDER BY COUNT(e.snum) DESC
LIMIT 1;
```

4. Find the number of unique students that every professor teaches.

   (a) Answer :

```
SELECT t.fid, COUNT(t.snum)
FROM
      (SELECT DISTINCT c.fid, e.snum
      FROM Enrolled e, Class c
      WHERE e.cname=c.name
      ORDER BY c.fid) t
GROUP BY t.fid;
```

5. Find the names of all Sophomore Computer Science majors who are enrolled in a class taught by Mark Sheldon.

   (a) Answer :

```
SELECT s.sname
FROM Student s
WHERE s.level=2
AND s.snum IN
      (SELECT e.snum
      FROM Enrolled e
      WHERE e.cname IN (
            SELECT c.name
            FROM Class c
            WHERE c.fid IN (
                  SELECT f.fid
                  FROM Faculty f
                  WHERE f.fname = "Mark Sheldon")));
```

6. Find the name of the youngest student who is an American Studies major or in an Intro to International Relations class.

   (a) Answer :

```
SELECT s.sname
FROM Student s
WHERE s.major="American Studies" OR s.snum IN
      (SELECT e.snum
```

```
        FROM Enrolled e
        WHERE e.cname="Intro to International Relations")
    AND s.age =
        (SELECT MIN(s.age)
        FROM Student s
        WHERE s.major="American Studies" OR s.snum IN
            (SELECT e.snum
            FROM Enrolled e
            WHERE e.cname="Intro to International Relations"));
```

7. Print the average level of students in each class, for every class.

   (a) Answer :

```
SELECT c.name, AVG(s.level)
FROM Student s, Class c, Enrolled e
WHERE e.snum = s.snum AND e.cname = c.name
GROUP BY c.name;
```

8. Print the average age and average level of students in each major, for every major.

   (a) Answer :

```
SELECT s.major, AVG(s.age), AVG(s.level)
FROM Student s
GROUP BY s.major;
```

9. Find the major in which the most students have more than one class with a given professor.

   (a) Answer :

```
SELECT q1.major
FROM
    (SELECT s.major, COUNT(*) AS num_students
     FROM Student s, Enrolled e1, Enrolled e2, Class c1, Class c2
     WHERE s.snum = e1.snum AND s.snum = e2.snum
        AND e1.cname = c1.name AND e2.cname = c2.name
        AND c1.fid = c2.fid
        AND c1.name < c2.name
     GROUP BY s.major) q1
WHERE q1.num_students =
    (SELECT MAX(q2.num_students)
     FROM
        (SELECT COUNT(*) AS num_students
         FROM Student s, Enrolled e1, Enrolled e2, Class c1, Class c2
         WHERE s.snum = e1.snum AND s.snum = e2.snum
            AND e1.cname = c1.name AND e2.cname = c2.name
            AND c1.fid = c2.fid
```

AND c1.name < c2.name
GROUP BY s.major) q2);

10. Find all pairs of students taking the same courses.

    (a) Answer :
    SELECT DISTINCT s1.sname, s2.sname
    FROM Student s1, Student s2, Enrolled e1, Enrolled e2
    WHERE s1.snum = e1.snum
    AND s2.snum = e2.snum
    AND e1.cname = e2.cname
    AND e1.snum < e2.snum

11. Find all pairs of students taking courses from the same professors.

    (a) Answer :
    SELECT DISTINCT s1.sname, s2.sname
    FROM Student s1, Student s2,
    Enrolled e1, Enrolled e2,
    Class c1, Class c2
    WHERE s1.snum = e1.snum
    AND s2.snum = e2.snum
    AND e1.cname = c1.name
    AND e2.cname = c2.name
    AND c1.fid = c2.fid
    AND s1.snum < s2.snum

12. For each major, find the student with the largest gap in their schedule (time between two classes).

    (Note: This is a challenge problem! The solution below finds, for each major, the student with the largest gap between *any* two classes, even if there is another class between them. Any solution along these lines will earn full credit.)

    (a) Answer :
    SELECT s.major, s.sname, (c2.time - c1.time) AS gap
    FROM Student s, Enrolled e1, Enrolled e2, Class c1, Class c2
    WHERE s.snum = e1.snum AND e1.snum = e2.snum
    AND e1.cname = c1.name AND e2.cname = c2.name
    AND c1.time < c2.time
    AND (s.major, (c2.time - c1.time)) IN
        (SELECT s.major, MAX(c2.time - c1.time) AS max_gap_for_major
        FROM Student s, Enrolled e1, Enrolled e2, Class c1, Class c2
        WHERE s.snum = e1.snum AND e1.snum = e2.snum

```
AND e1.cname = c1.name AND e2.cname = c2.name
AND c1.time < c2.time
GROUP BY s.major);
```