Midterm Test—Sample Solutions

CSC343H1 / LEC0101 —Winter 2020 —
Danny Heap

February 26, 12:10/1:10 p.m. — Duration: **50 minutes**

Question 1. [5 marks]

Indicate whether each statement is **True** or **False** by circling the appropriate answer.

Part (a) [1 mark]

For any relation R, with a subset of R's attributes A, and condition C, the query $\Pi_A(\sigma_C R)$ never gives the same result as $\sigma_C(\Pi_A R)$.

True **False**

sample solution: **False**, just let C be 1 = 1.

Part (b) [1 mark]

A valid relational algebra query always returns more rows than columns.

True **False**

sample solution: **False**, there are certainly queries that return a single multi-attribute tuple.

Part (c) [1 mark]

A natural join of two relations always produces a relation with at least as many attributes as either of the two original relations.

True **False**

sample solution: **true**, the joined relation has the union of the two sets of attributes.

Part (d) [1 mark]

A theta join of two relations never produces a relation with more attributes than the maximum number of attributes in the two original relations.

True **False**

sample solution: **False**, it produces a relation with the sum of the number of attributes in the original relations

Part (e) [1 mark]

A relational algebra query that uses the assignment operator can never be replaced by an equivalent query that doesn't use the assignment operator. **True** **False**

sample solution: **False**,

Question 2. [10 MARKS]

Recall this relational algebra schema from class:

Student(<u>sID</u>, surName, firstName, campus, email, cgpa)
Course(<u>dept, cNum</u>, name, breadth)
Offering(<u>oID</u>, dept, cNum, term, instructor)
Took(<u>sID, oID</u>, grade)

Offering[dept, cNum] ⊆ Course[dept, cNum]
Took[sID] ⊆ Student[sID]
Took[oID] ⊆ Offering[oID]

Part (a) [5 MARKS]

Consider the following constraint:

$$\text{Soff(s,c,g)} := \Pi_{sID,cNum,grade}(\sigma_{dept='CSC'}(\rho_T \text{Took} \bowtie_{T.oID=O.oID} \rho_O \text{Offering}))$$

$$\text{2Soff(s,c,g)} := \Pi_{S1.s,S1.c,S1.g}(\sigma_{S1.g<S2.g \wedge S1.s=S2.s \wedge S1.c=S2.c}(\rho_{S1}\text{Soff} \times \rho_{S2}\text{Soff}))$$

$$\sigma_{Q.s=R.s \wedge Q.c=R.c \wedge Q.g<R.g}(\rho_Q \text{Soff} \times \rho_R \text{2Soff}) = \varnothing$$

Define instances of Took and Offering that violate the constraint **only**:

Took:

sID	oID	grade
666	777	65
666	888	75
666	999	64

Offering:

oID	dept	cNum	term	instructor
777	'CSC'	343	20189	'Horton'
888	'CSC'	343	20191	'Horton'
999	'CSC'	343	20199	'Horton'

Midterm Test—Sample Solutions

CSC343H1 / LEC0101 —Winter 2020 —
Danny Heap

February 26, 12:10/1:10 p.m. — Duration: **50 minutes**

Part (b) [5 marks]

Write a relational algebra query to find the first name and surname of student(s) who took two different CSC courses, in two different terms, where their grades differed by more than 10 percent. Break up your query into steps, using the assignment operator to create renamed relations, together with their attributes, on the left-hand side. Adding commentary will help you (and the grader!) understand your answer.

sample solution:

// sid, grade, cnum, term of CSC students

$$\text{CSC-Taker(s,g,c,t)} := \Pi_{sid,grade,cnum,term}[\sigma_{dept='CSC'}(\text{Took} \bowtie \text{Offering})]$$

// sids of those with over 10 grade spread, different courses, different terms

$$\text{GradeSpreadTakers(s)} := \Pi_{sid}[\sigma_{S1.s=S2.s \wedge S1.c<S2.c \wedge S1.t<>S2.t \wedge |S1.grade-S2.grade|>10}(\rho_{S1}\text{CSC-Taker} \times \rho_{S2}\text{CSC-Taker})]$$

// names and surnames

$$\Pi_{name,surname}(\text{GradeSpreadTakers} \bowtie \text{Student})$$

Student(<u>sID</u>, surName, firstName, campus, email, cgpa)
Course(<u>dept, cNum</u>, name, breadth)
Offering(<u>oID</u>, dept, cNum, term, instructor)
Took(<u>sID, oID</u>, grade)

Offering[dept, cNum] ⊆ Course[dept, cNum]
Took[sID] ⊆ Student[sID]
Took[oID] ⊆ Offering[oID]

Question 3. [7 marks]

For this question use the university schema, below:

Student(<u>sID</u>, surName, firstName, campus, email, cgpa)
Course(<u>dept, cNum</u>, name, breadth)
Offering(<u>oID</u>, dept, cNum, term, instructor)
Took(<u>sID, oID</u>, grade)

Offering[dept, cNum] ⊆ Course[dept, cNum]
Took[sID] ⊆ Student[sID]
Took[oID] ⊆ Offering[oID]

Part (a) [4 marks]

Write an SQL query to find students who took at least 1 CSC course, but never took the same CSC course in two different terms. Report the sID and minimum grade in CSC courses taken by these students. Organize your output in ascending order of the minimum grade in CSC courses taken.

```
csc343=> create view csc_takers as
select sid, cnum, term
from took, offering
where took.oid = offering.oid;

csc343=> create view double_takers as
select c1.sid
from csc_takers c1, csc_takers c2
```

```
where c1.sid = c2.sid
and c1.cnum = c2.cnum
and c1.term < c2.term;

csc343=> create view single_takers as
(select sid from csc_takers)
except
(select sid from double_takers);

csc343=> select s.sid, min(grade)
from single_takers s, took t, offering o
where s.sid = t.sid
and t.oid = o.oid
and dept='CSC'
group by s.sid
order by min(grade);
  sid  | min
-------+-----
 99132 |  79
(1 row)
```

Midterm Test—Sample Solutions

CSC343H1 / LEC0101 —Winter 2020 —
Danny Heap

February 26, 12:10/1:10 p.m. — Duration: **50 minutes**

The following query is supposed to print the sid and average grade of the student(s) who has the highest average grade. It runs but does not always give the correct output.

```
SELECT student.sid, avg(took.grade)

FROM student NATURAL JOIN took

GROUP BY student.sid

HAVING avg(grade) >= ANY

  (SELECT avg(grade) from took group by sid);
```

Part (b) [1 mark]

Generalizing to any dataset, explain what is wrong with the output of this query.

Solution: It gives all average grades of all students.

Part (c) [1 mark]

Explain (in plain English) the property of the datasets that would cause this query to produce the **correct** result. That is, what must be true about the datasets such that the query would produce the correct result (you cannot say 'empty relations')?

Solution: At most 1 student took courses.

Part (d) [1 mark]

Fix the query by making the smallest change that you can. Write your corrections directly on the query text above.

Solution:

```
change ANY to ALL
```

schema repeated below for convenience.

Relations

Student(<u>sID</u>, surName, firstName, campus, email, cgpa)
Course(<u>dept, cNum</u>, name, breadth)
Offering(<u>oID</u>, dept, cNum, term, instructor)
Took(<u>sID, oID</u>, grade)

Integrity constraints

Offering[dept, cNum] ⊆ Course[dept, cNum]
Took[sID] ⊆ Student[sID]
Took[oID] ⊆ Offering[oID]

Question 4. [4 MARKS]

Suppose we have the following tables from a Twitter database:

```
Follows:
     a     |       b
-----------+----------------
 sina      | kanyewest
 sina      | RonConwayFacts
 diane     | LilaFontes
 diane     | swcarpentry
 diane     | mfeathers
 diane     | sina
 michelle  | sina
 michelle  | diane
 michelle  | Jeff
(9 rows)
```

```
Profile:
  userid  |    name    | location
----------+------------+----------
 alan     | catman     | Ottawa
 sina     | superman   |
 diane    | superwoman | Toronto
 michelle | rockstar   | Montreal
(4 rows)
```

```
Tweets:
 id  | userid |    content
-----+--------+----------------
 123 | alan   | hello twitter
 125 | alan   | bye twitter
 126 | alan   | hello twitter
 128 | alan   | bye twitter
 476 | sina   | hello twitter
 553 | diane  | hello twitter
(6 rows)
```

Show the result of running each of the following queries. If a table is produced, include the column names. If the query generates an error, explain. If any null values occur, write them out clearly as `NULL`.

Midterm Test—Sample Solutions

CSC343H1 / LEC0101 —Winter 2020 —
Danny Heap

February 26, 12:10/1:10 p.m. — Duration: **50 minutes**

Solutions

```
SELECT p.userid, t.content, count(p.location)
FROM profile p, tweets t
WHERE t.userid = p.userid
GROUP BY p.userid, t.content;


-- Output:

 userid |    content     | count
--------+----------------+-------
 alan   | hellow twitter |     2
 diane  | hellow twitter |     1
 alan   | bye twitter    |     2
 sina   | hellow twitter |     0
(4 rows)
```

```
(select id from profile natural join tweets)
UNION ALL
(select userid from profile where name='catman'

-- Output:

ERROR:  UNION types integer and character
varying cannot be matched
LINE 3: (select userid from profile
where name='catman');
```