Recall this schema, which we have used many times in class.

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] $\subseteq$ Course[dept, cNum]

Took[sID] $\subseteq$ Student[sID]

Took[oID] $\subseteq$ Offering[oID]

Question 1. [5 MARKS]

Part (a) [1 MARK]

According to the schema, can the same cNum be used by two different departments for offerings that occur in the same term? [Yes] No.

Part (b) [2 MARKS]

Consider this constraint:

$$Proom(cNum1, cNum2, term) :=$$

$$\Pi_{O1.cNum, O2.cNum, O1.term}\sigma_{\substack{O1.cNum<O2.cNum \\ \wedge \\ O1.dept=O2.dept='CSC' \\ \wedge \\ O1.instructor=O2.instructor \\ \wedge \\ O1.term=O2.term}}[(\rho_{O1}Offering) \times (\rho_{O2}Offering)]$$

$$\sigma_{\substack{P1.cNum1=P2.cNum1 \\ \wedge \\ P1.cNum2=P2.cNum2 \\ \wedge \\ P1.term\neq P2.term}}[(\rho_{P1}Proom) \times (\rho_{P2}Proom)] = \emptyset$$

Define an instance of Offering that violates the constraint.

Solution:

oID	dept	cNum	term	instructor
o1	CSC	343	termA	Sina
o2	CSC	443	termA	Sina
o3	CSC	343	termB	Diane
o4	CSC	443	termB	Diane

Part (c) [2 MARKS]

Write the following constraint using relational algebra: csc343 can be offered at the same time as csc443 only if prof Horton teaches both courses.

Solution:

$$\sigma_{01.term=O2.term \wedge O1.cNum='443' \wedge 02.cNum='343' \wedge (O1.dept=O2.dept='CSC') \wedge (01.instructor \neq 'Horton' \vee 02.instructor \neq 'Horton')} \\ [(\rho_{O1} Offering) \times (\rho_{O2} Offering)] = \emptyset$$

Question 2. [8 MARKS]

Write a query in relational algebra to find the sID of every student who has taken exactly one CSC course but has never taken a breadth course (a course where 'breadth' is true).

Solution: this is one possible way to solve this

– This student has taken a CSC course.

$Taken(sID, cNum) := \Pi_{(sID,cNum)}\sigma dept =' CSC'[Offering \bowtie Took]$

– This student has taken more than 1 CSC course.

$MoreThanOnce(sID) :=$

$$\Pi_{T1.sID}\sigma_{T1.sID=T2.sID \wedge T1.cNum \neq T2.cNum}[(\rho_{T1}Taken) \times (\rho_{T2}Taken)]$$

– This student has taken exactly 1 CSC course.

$ExactlyOnce(sID) :=$

$$\Pi_{sID}(Taken) - MoreThanOnce$$

– All offerings of breath courses.

$Breadth(oID) := \Pi_{oID}\sigma_{breadth=true}(Course \bowtie Offering)$

–Students who have taken Breath courses

$Breathers(sID) := \Pi_{sID}(Breadth \bowtie Took)$

–Answer

$Answer := ExactlyOnce - Breathers$

Question 3. [5 MARKS]

Suppose we want to find the campus of the student with the highest cgpa. If there was a tie, the campus name of all the tied campuses should be reported.

The following query attempts to solve this. It is syntactically correct, but doesn't always produce the right answer.

$Campusgpas(campus, cgpa) := \Pi_{campus,cgpa} Student$

$Nothighest(campus) := \Pi_{C1.campus} \sigma_{C1.cgpa<C2.cgpa}[(\rho_{C1} Campusgpas) \times (\rho_{C2} Campusgpas)]$

$Answer(campus) := (\Pi_{campus} Campusgpas) - Nothighest$

Part (a) [2 MARKS]

Define an instance of Student on which the query gives the wrong answer.

Solution:

sID	surName	firstName	campus	email	cgpa
1	Meraji	Sina	STG	s.m@gmail.com	3.4
2	Horton	Diane	STG	h.d@gmail.com	3.7

What relation *should be* produced in this case?

Solution:

campus
STG

What relation *is* produced in this case?

Solution:

campus

Part (b) [3 MARKS]

On the query above, make the smallest change(s) that will correct it.

Solution: Add cGPA to the attributes of Nothighest. It will need to be added to both the left-hand side and to the project statement. Then do the subtraction in the RHS of answer before the project.

Question 4. [7 MARKS]

Consider this new schema for a music industry database:

Relations

Musician(<u>mID</u>, surName, firstName, birthdate)

Album(<u>aID</u>, title, mID, year)

RecordCompany(<u>cID</u>, name, president)

Produced(<u>aID, cID</u>)

Integrity constraints

Album[mID] $\subseteq$ Musician[mID]

Produced[aID] $\subseteq$ Album[aID]

Produced[cID] $\subseteq$ RecordCompany[cID]

Part (a) [1 MARK]

Suppose relation Produced has 200 tuples. How many tuples could RecordCompany have? Circle all that apply:

Solution:

0 [1] [179] [200] [201]

Part (b) [2 MARKS]

Which of the following are true according to the schema? Circle Yes or No for each.

An album can be produced by 2 different companies	[Yes]	No
An album can have more than one musician	Yes	[No]
Every record company has produced at most one album.	Yes	[No]
Every album has at most one record company that produced it.	Yes	[No]
A person cannot be president of 2 record companies.	Yes	[No]

Part (c) [2 MARKS]

Write a constraint in relational algebra that says the name of each record company must be the same as the surname of some musician.

Solution:

$\pi_{name}(RecordCompaney) - \pi_{surName}(Musician) = \emptyset$

Part (d) [2 MARKS]

Suppose every album is produced in one location, and we add an attribute named 'location' to the Produced relation to keep track of it. Is it a good idea to store this 'location' information in the Produced relation? Circle one: **Solution:**

Yes [No].

The same album can have multiple rows in Produced. The location information will be repeated in all of them. This is unnecessary duplication. The information should go in the Album relation.

Question 5. [8 MARKS]

For this question, you will write SQL queries using a simplified version of the Instagram schema from Assignment 1. (We removed unnecessary pieces and renamed one table and one attribute because `user` and `when` are reserved words in SQL.)

Relations

Account(uID, name, website, phone)

Follows(follower, followed)

Post(pid, uid, location, caption)

Comment(pid, commenter, comment_time, text)

Hashtag(pid, tag)

Integrity constraints

Follows[follower] $\subseteq$ Account[uID]

Follows[followed] $\subseteq$ Account[uID]

Post[uID] $\subseteq$ Account[uID]

Comment[pID] $\subseteq$ Post[pID]

Comment[commenter] $\subseteq$ Account[uID]

Hashtag[pID] $\subseteq$ Post[pID]

$\sigma_{follower=followed} Follows = \emptyset$

Part (a) [3 MARKS]

In our schema, users can comment multiple times on the same post and the comment text can be null. Write a query in SQL to find, for each user who has made an actual comment (where the text is not null), their name and the number of posts they have made an actual comment on. Report the user's name and the number of posts. Organize the output in non-increasing order by the number of posts.

Solution:

```
SELECT name, count (DISTINCT pid)
FROM Comment, Account
WHERE uid = commenter
AND text IS NOT NULL
GROUP BY uid
ORDER BY count(DISTINCT pid)  DESC;
```

Part (b) [2 MARKS]

Write a query in SQL that finds the uid, with no duplicates, of people who have made posts but have never used a hashtag.

Solution:

```
(select uid from post) except (select uid from post natural join hashtag);
```

The following query is supposed to print the number of pairs of mutual followers, that is, users who follow each other. It runs but does not always give the correct output.

```
SELECT count(*)

FROM (

    SELECT F1.follower, F2.follower

    FROM  Follows F1, Follows F2

    WHERE F1.follower = F2.followed

    AND F2.follower = F1.followed

) AS MutualFollowers;
```

Part (c) [1 MARK]

Suppose that Follows has these values. What will be the output of the query?

```
 follower | followed
----------+----------
        2 |        1
        3 |        2
        2 |        3
        4 |        2
        1 |        3
```

Solution:

```
 count
-------
     2
```

Part (d) [1 MARK]

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

Solution: It counts every pair twice.

Part (e) [1 MARK]

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

Solution: One option is:

```
add    AND F1.followed < F2.followed
```

Another solution is to do count(*)/2

Question 6. [8 MARKS]

Suppose we have the following tables:

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

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

```
Tweets:
id   | userid |    content
-----+--------+----------------
 123 | alan   | hellow twitter
 125 | alan   | bye twitter
 126 | alan   | hellow twitter
 128 | alan   | bye twitter
 476 | sina   | hellow twitter
 553 | diane  | hellow 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.

Solutions

```
SELECT count(*)
FROM Profile RIGHT JOIN Follows
    ON a = id;


-- Output:

 count
-------
    9
(1 row)
```

```
Select P.id, count(Follows.b) AS followers
From Profile P join Follows ON P.ID=Follows.b
AND P.location='Toronto' Group by(P.ID);
-- Output:

   id    | followers
---------+-----------
 diane   |         1
(1 row)
```

Here are the tables again, for easy reference:

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

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

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

```
SELECT P.id, count(T.content) AS number
FROM Profile P JOIN  Tweets t
    On T.userid = P.id
AND P.location='Montreal';
```

```
-- Output:

ERROR:  column "p.id" must appear in the GROUP BY
clause or be used in an aggregate function
LINE 1: SELECT P.id, count(T.content) AS number
```

```
Select Tweets.content
From Tweets Join Profile
On Tweets.userid = Profile.ID
And Profile.location IN
(select location from profile
 where name='catman');
```

```
    content
----------------
 hellow twitter
 bye twitter
 hellow twitter
 bye twitter
(4 rows)
```