CSCC43 UTSC

Assignment 1

50 points

Due Date: June 8, 11:59pm

Learning Goals

By the end of this assignment, you should be able to:

1. Read a new relational schema, and determine whether a particular instance is valid with respect to that schema,

2. Apply the individual techniques for writing relational algebra queries and integrity constraints that we learned in class,

3. Combine the individual techniques to solve complex problems, and

4. identify problems that cannot be solved using relational algebra.

These skills we leave you well prepared to learn SQL.

Instructions

There are two schemas in this assignment and queries related to those schemas. You need to answer all the questions.

- Schema 1 has 5 queries (40%) and 2 integrity constraint (10%).
- Schema 2 has 5 queries (50%) and no integrity constraint.

Common Instructions for both schema:

Write the queries below in relational algebra. There are several variations on relational algebra, and different notations for the operations. You must use the same notation as we have used in class and on the slides. You may use assignment, and the operators we have used in class: π, σ, $\bowtie$, $\bowtie_{\theta}$, $\times$, $\cap$, $\cup$, $-$, ρ. Assume that all relations are sets (not bags), as we have done in class, and **do not use any of the extended relational algebra** operations from Chapter 5 of the textbook (for example, do not use the division operator). Some additional points to keep in mind:

- Do not make any assumptions about the data that are not enforced by the original constraints given above, including the ones written in English. Your queries should work for any database that satisfies those constraints.
- Assume that every tuple has a value for every attribute. For those of you who know some SQL, in other words, there are no null values.
- Remember that the condition on a select operation may only examine the values of the attributes in one tuple, not whole columns. In other words, to use a value (other than a literal value such as 100 or "Adele"), you must get that value into the tuples that your select will examine.

- The condition on a select operation can use comparison operators (such as ≤ and ≥) and Boolean operators (∨, ∧ and ¬). Simple arithmetic is also okay,
  e.g., attribute1≤ attribute2 + 5000.
- Some relations in our schema have a date-time attribute. You may use comparison operators on such values. You may refer to the year component of a date-time attribute d using the notation d.year.
- You are encouraged to use assignment to define intermediate results.
- It's a good idea to add commentary explaining what you're doing. This way, even if your final answer is not completely correct, you may receive part marks.
- The order of the columns in the result does not matter.
- When asked for a maximum or minimum, if there are ties, report all of them.

At least one of the queries cannot be expressed in the language that you are using. In those cases, simply write "cannot be expressed".

Schema 1

Rather than store pictures and videos themselves, our database will store URL references to their locations in the cloud.

Relations

- User (uid, name, website, about, email, phone, photo)

A tuple in this relation represents an Instagram user. uid is the string identifier selected by the user. $name, website, about, email,$ and $phone$ are information about this user. $photo$ is the url of the profile photo of this user.

- Follows (follower, followed, start)

A tuple in this relation represents the fact that the user with identifier $follower$ follows the user with the identifier $followed$, beginning at date-time $start$.

- Post (pid, uid, when, location, caption)

A tuple in this relation represents a post added by a user to their profile. pid is the post identification number. uid is the identification number of the user who posted this post, which we will call the poster. $when$ is the date-time when this post was posted by the poster. $location$ is the location selected by the poster for this post. $caption$ is the text description given to this post by the poster. Each post can have at most one caption.

- PIncludes (pid, url)

A tuple in this relation represents the fact that the photo or video stored at location url is included in post pid.

- Hashtag (pid, tag)

A tuple in this relation represents the fact that the caption of post pid includes the hashtag tag.

- Likes (<u>liker</u>, <u>pid</u>, when)

A tuple in this relation represents the fact that user $liker$ has liked the post pid at date-time $when$.

- Comment (<u>pid</u>, <u>commenter</u>, <u>when</u>, text)

A tuple in this relation represents the fact that user $commenter$ left the comment $text$ for post pid at date-time $when$.

- Story (<u>sid</u>, uid, when, current)

A tuple in this tuple represents a story created by a user. sid is the identifier of the story, uid is the user who created it, and $when$ is the date-time when they created it. $current$ is true iff this is the current story for this user.

- SIncludes (<u>sid</u>, <u>url</u>)

A tuple in this relation represents the fact that the photo or video stored at location url is included in story sid.

- Saw (<u>viewerid</u>, <u>sid</u>, when)

A tuple in this relation represents the fact that viewer $viewerid$ saw story sid at date-time $when$.

Integrity Constraint

- Follows[follower] ⊆ User[uid]
- Follows[followed] ⊆ User[uid]
- Post[uid] ⊆ User[uid]
- PIncludes[pid] ⊆ Post[pid]
- Hashtag[pid] ⊆ Post[pid]
- Likes[likerid] ⊆ User[uid]
- Likes[pid] ⊆ Post[pid]
- Comment[pid] ⊆ Post[pid]
- Comment[commenter] ⊆ User[uid]
- Story[uid] ⊆ User[uid]
- SIncludes[sid] ⊆ Story[sid]
- Saw[viewerid] ⊆ User[uid]
- Saw[sid] ⊆ Story[sid]
- $\sigma_{follower=followed}Follows = \emptyset$
- Story[current] ⊆ $\{“yes”, “no”\}$

Warmup: Getting to know the schema

To get familiar with the schema, ask yourself questions like these (but don't hand in your answers):

- What does this integrity constraint mean? $\sigma_{follower=followed}Follows = \emptyset$
- Would it be a good idea to define the Follows relation like this?
  Follows (follower, followed, start)
- Can the database represent a single post that has multiple comments?
- Can the database represent multiple comments from the same user on one post?
- How does the schema allow any number of photos or videos to be included in one story, but restrict the user to having only one profile photo?
- Can the database represent that a user likes the same post more than once? (If not, how would one change the schema to allow this?)
- Can the database represent that a user makes two posts at the same time?
- Can the database represent that the same user makes the same comment on two different posts?
- Can the database represent that the same picture is included in 2 stories?

Part 1: Queries (40% - 4 points each)

Note: The queries are not in order according of difficulty.

1. Find all the users who have never liked or viewed a post or story of a user that they do not follow. Report their user id and "about" information. Put the information into a relation with attributes "username" and "description".
2. Find every hashtag that has been mentioned in at least three post captions on every day of 2020. You may assume that there is at least one post on each day of a year.
3. Let's say that a pair of users are "reciprocal followers" if they follow each other. For each pair of reciprocal followers, find all of their "uncommon followers": users who follow one of them but not the other. Report one row for each of the pair's uncommon follower. In it, include the identifiers of the reciprocal followers, and the identifier, name and email of the uncommon follower.
4. Find the user who has liked the most posts. Report the user's id, name and email, and the id of the posts they have liked. If there is a tie, report them all.
5. Let's say a pair of users are "backscratchers" if they follow each other and like all of each other's posts. Report the user id of all users who follow some pair of backscratcher users.

Part 2: Additional Integrity Constraints (20% - 2.5 points each)

Express the following integrity constraints with the notation R=∅, where R is an expression of relational algebra. You are welcome to define intermediate results with assignment and then use them in an integrity constraint.

1. Each user can have at most one current story.
2. Every post must include at least one picture or one video and so must every story.

When writing your queries, do not assume that these additional integrity constraints hold, except for the second one — it was described above as a constraint that holds.

Schema 2

For this problem we use the schema of the Health Network database. It includes the following relations. Keys are underlined and each tuple is described.

- Clinic (clinicID, hName, hAddress, hCity)

  A clinic including the $clinic\ ID$, $name$, $address$ and $city$ of that clinic.

- Room (roomID, clinicID, rNumber, rSize)

  A clinic room including the clinic ID, room number, and room capacity – the number of patients it accommodates.

- Employee (employeeID, eFirstName, eLastName, eRole, eSalary, eAddress, eCity, eSIN)

  An employee including their employee ID number, first and last names, role in the clinic (or occupation type, such as physician, technician, etc.), salary, address, city, and Social Insurance Number.

- Department (departmentID, clinicID, dName, dHead)

  A department in a clinic including the clinic ID, the department ID, the name of the department, and the employee ID of the head of the department.

- EmployeeDepartment (employeeID, departmentID, edStartDate, edEndDate)
  The departments to which an employee belongs. An employee may work in more than one department. This tuple contains the employee ID, the department ID, and the dates that they started and finished employment in that department.

- Patient (patientID, pFirstName, pLastName, pSex, pOHIP, pOther, pDOB, pAddress, pCity, pProvince, pCountry)

  A patient in the clinic system including the patient ID, first name, last name, sex (M or F), OHIP number, other form of payment if not on OHIP, date of birth, residence address including the address, city, province, and country.

- PatientAppointment (patientID, departmentID, employeeID, paDate, paTime, paReason)

  An appointment for a patient, including the department ID of the clinic department, the employee ID of the person whom they are seeing (could be technician), the date, the time, and the purpose of the visit.

- PatientRoomAssignment (patientID, roomID, prInDate, prOutDate, prPhysician)

A room that is assigned to a patient who needs to stay in the clinic - including the ID of the patient and room, the date that they were admitted, the date that they left the clinic, and the employee ID of their attending physician during their stay.

Integrity Constraint

- Room (clinicID) ⊆ Clinic (clinicID)
- Department (clinicID) ⊆ Clinic (clinicID)
- Department (dHead) ⊆ Employee (employeeID)
- EmployeeDepartment (departmentID) ⊆ Department (departmentID)
- PatientAppointment (patientID) ⊆ Patient(patientID)
- PatientAppointment (departmentID) ⊆ Department (departmentID)
- PatientAppointment (employeeID) ⊆ Employee (employeeID)
- PatientRoomAssignment (patientID) ⊆ Patient (patientID)
- PatientRoomAssignment (roomID) ⊆ Room (roomID)

Queries: (50% - 5 points each)

1. Find the clinic name and department name for all departments whose head is also head of another department.
2. Find the first name and last name of all Physicians whose patients are never in the clinic for less than 10 days.
   (They must have had patients admitted in the clinic at some time.)
3. Find the first and last name of all patients who have exactly two physicians in the clinic's system.
   (Note that physicians are specified in both appointments and room assignment.)
4. Find the clinic and department name for the department at which there were no appointments from April 3, 2019 to April 8, 2019.
5. Find the clinic, room number, and patient first and last names of all semi-private (size = 2) rooms which were occupied by female patients who were over 50 on May 13, 2019.

Style and formatting requirements

To make your algebra more readable, and to minimize errors, we are including these style and formatting requirements:

- In your assignment statements, you must include names for all attributes in the intermediate relation you are defining.

For example, write

$$Highest\ Grade(sID, oID, grade) :=\ldots$$

- Use meaningful names for intermediate relations and attributes, just as you would in a program.
- If you want to include comments, put them before the algebra that they pertain to, not after. Make them stand out from the algebra, for example by using a different font.

  For example, this looks reasonable:–

  Students who had very high grades in any offering of a csc

  $$course.\, High(sID) := \pi_{sID \sigma dept='csc' \wedge grade > 95}(Took \bowtie Offering)$$

  A modest portion of your mark will be for good style and formatting.

Submission Instructions:

Your assignment must be typed; handwritten assignments will not be marked. You may use any word-processing software you like. Many academics use LaTeX. It produces beautifully typeset text and handles mathematical notation well. If you would like to learn LaTeX, there are helpful resources online.

Whatever you choose to use, you need to produce a final document in pdf format. You must declare your team (whether it is a team of one or two students) and hand in your work electronically using the MarkUs online system.

Well before the due date, you should declare your team and try submitting with MarkUs.

For this assignment, hand in just one file: **A1.pdf**.

If you are working in a pair, only one of you should hand it in. Check that you have submitted the correct version of your file by downloading it from MarkUs; new files will not be accepted after the due date.