University of Toronto
CSC343, Fall 2017

Assignment 1

Due: Wed Oct 11, 8pm

Learning Goals

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

1. read a new relational schema, and determine whether or not 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.

Later in the course, you will learn about how to develop your own schema based on knowledge of the domain. Even though developing a schema is really the first step in building a database, it is a more advanced task than querying an existing database; this is why we will be learning about it and practising it later.

Domain

For this assignment, you will operate on a database for Instagram. Instagram is a social media platform on which users share pictures and videos, and post comments about them. You need to know a few facts about the domain:

- As on many social media platforms, a user may **follow** another user.
- The follows relationship is not necessarily symmetric: x can follow y even if y doesn't follow x. If x follows y, we say that x is a **follower** of y, and that y is **followed by** x.
- A **post** has one or more photos and/or videos. Users can **like** posts, and write comments on them. Their comments may include hashtags (for example, #the6ix, #uoft).
- A user can create **stories**. Like a post, a story has one or more photos and/or videos, but it has no hashtags and can't be "liked". Instagram does, however, keep track of the users that view a story.
- A user has the option of having a single **current story** at any one time. Their current story is displayed prominently when other users view their profile.

The above may or may not be completely accurate for Instagram; regardless, our schema and your queries will be based on it.

Schema

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(liker, pid, when)

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

- Comment(pid, commenter, when, text)

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

- Story(sid, 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(sid, url)

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

- Saw(viewerid, sid, when)

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

Integrity constraints

- Follows[follower] $\subseteq$ User[uid]
- Follows[followed] $\subseteq$ User[uid]
- Post[uid] $\subseteq$ User[uid]

- PIncludes[pid] $\subseteq$ Post[pid]
- Hashtag[pid] $\subseteq$ Post[pid]
- Likes[liker_id] $\subseteq$ User[uid]
- Likes[pid] $\subseteq$ Post[pid]
- Comment[pid] $\subseteq$ Post[pid]
- Comment[commenter] $\subseteq$ User[uid]
- Story[uid] $\subseteq$ User[uid]
- SIncludes[sid] $\subseteq$ Story[sid]
- Saw[viewerid] $\subseteq$ User[uid]
- Saw[sid] $\subseteq$ Story[sid]
- $\sigma_{follower=followed} Follows = \emptyset$
- Story[current] $\subseteq$ {"*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(<u>follower</u>, <u>followed</u>, 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

Write the queries below in relational algebra. There are a number of 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: $\Pi, \sigma, \bowtie, \bowtie_{condition}, \times, \cap, \cup, -, \rho$. 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 $\leq$ and $\neq$) and boolean operators ($\vee$, $\wedge$ and $\neg$). Simple arithmetic is also okay, *e.g.*, attribute1 $\leq$ 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 doesn't 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". Note: The queries are not in order according to 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 2017. 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 others' posts. Report the user id of all users who follow some pair of backscratcher users.
6. The "most recent activity" of a user is his or her latest story or post. The "most recently active user" is the user whose most recent activity occurred most recently.

   Report the name of every user, and for the most recently active user they follow, report their name and email, and the date of their most-recent activity. If there is a tie for the most recently active user that a user follows, report a row for each of them.

7. Find the users who have always liked posts in the same order as the order in which they were posted, that is, users for whom the following is true: if they liked n different posts (posts of any users) and

$$[post\_date\_1] < [post\_date\_2] < ... < [post\_date\_n]$$

where $post\_date\_i$ is the date on which a post i was posted, then it holds that

$$[like\_date\_1] < [like\_date\_2] < ... < [like\_date\_n]$$

where $like\_date\_i$ is the date on which the post i was liked by the user. Report the user's name and email.

8. Report the name and email of the user who has gained the greatest number of new followers in 2017. If there is a tie, report them all.

9. For each user who has ever viewed any story, report their id and the id of the first and of the last story they have seen. If there is a tie for the first story seen, report both; if there is a tie for the last story seen, report both. This means that a user could have up to 4 rows in the resulting relation.

10. A comment is said to have either positive or negative sentiment based on the presence of words such as "like," "love," "dislike," and "hate." A "sentiment shift" in the comments on a post occurs at moment m iff all comments on that post before m have positive sentiment, while all comments on that post after m have negative sentiment — or the other way around, with comments shifting from negative to positive sentiment.

    Find posts that have at least three comments and for which there has been a sentiment shift over time. For each post, report the user who owns it and, for each comment on the post, the commenter's id, the date of their comment and its sentiment.

    You may assume there is a function, called *sentiment* that can be applied to a comment's text and returns the sentiment of the comment as a string with the value "positive" or "negative". For example, you may refer to $sentiment(text)$ in the condition of a select operator.

Part 2: Additional Integrity Constraints

Express the following integrity constraints with the notation $R = \emptyset$, 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. A comment on a post must occur after the date-time of the post itself. (Remember that you can compare two date-time attributes with simple <, >= etc.)

2. Each user can have at most one current story.

3. Every post must include at least one picture or one video and so must every story.

When writing your queries for Part 1, don't assume that these additional integrity constraints hold, except for the second one — it was described above as a constraint that holds.

Style and formatting requirements

In order 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

$$HighestGrade(sID, oID, grade) := \quad \ldots$$

- 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. Instructions for doing so are posted on the Assignments page of the course website. Well before the due date, you should declare your team and try submitting with MarkUs. You can submit an empty file as a placeholder, and then submit a new version of the file later (before the deadline, of course); look in the "Replace" column.

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.