University of Toronto
CSC343, Fall 2017

Assignment 1: James Leja 1002527571 Chris Stalzer - need Chris's student

Unary operators on relations:

- $\Pi_{x,y,z}(R)$
- $\sigma_{condition}(R)$
- $\rho_{New}(R)$
- $\rho_{New(a,b,c)}(R)$

Binary operators on relations:

- $R \times S$
- $R \bowtie S$
- $R \bowtie_{condition} S$
- $R \cup S$
- $R \cap S$
- $R - S$

Logical operators:

- $\vee$
- $\wedge$
- $\neg$

Assignment:

- $New(a, b, c) := R$

Stacked subscripts:

- $\sigma_{this.something>that.something \wedge \\ this.otherthing \leq that.otherthing}$

Below is the text of the assignment questions; we suggest you include it in your solution. We have also included a nonsense example of how a query might look in LaTeX. We used `\var` in a couple of places to show what that looks like. If you leave it out, most of the time the algebra looks okay, but certain words, *e.g.*, "Offer" look horrific without it.

The characters "\\" create a line break and "[5pt]" puts in five points of extra vertical space. The algebra is easier to read with extra vertical space. We chose "–" to indicate comments, and added less vertical space between comments and the algebra they pertain to than between steps in the algebra. This helps the comments visually stick to the algebra.

Part 1: Queries

1. Find all the users who have never sent a message, but who have been sent at least one message. The message may have been sent to the user or to a group that the user belongs to. Report each user id.

   – Messages that have been sent.

   $hasSent(mID, uID, status) := \sigma_{status='sent'}Status$

   – uID of users who have received a message.

   $hasReceived(uID) := \Pi_{uID}(Status - hasSent)$

   – uID of users who have never sent a message.

   $neverSent(uID) := \Pi_{uID}User - \Pi_{uID}hasSent$

   $Q1(uID) := neverSent \cap hasReceived$

   – This is just an example for if we need two lines. The above slashes also create newlines.
   $neverSent(uID) :=$

   $$\Pi_{T1.sID}\sigma_{T1.oID \neq T2.oID \wedge T1.sID=T2.sID \wedge T1.grade=100 \wedge T2.grade=100}[(\rho_{T1}Took) \times (\rho_{T2}Took)]$$

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.