University of Toronto
CSC343, Fall 2017

Assignment 1:

First Name: Fenglun

Last Name: Wu

Student ID: 1002596684

First Name: Qiangyu

Last Name: Zheng

Student ID: 1002144128

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 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".

   – All the likers and the corresponding poster's uid.

   $LikerAndPoster(liker, uid) :=$

   $$\Pi_{liker,uid}[(Post \bowtie Likes)]$$

   – All the likers who have liked a post and followed the corresponding poster.

   $LikerFollowPoster(liker, uid) :=$

   $$\Pi_{liker,uid}\sigma_{liker=follower \wedge uid=followed}[(LikerAndPoster \times Follows)]$$

   – All the likers who have liked a post and not followed the corresponding poster.

   $LikerNotFollow(liker) :=$

   $$\Pi_{liker}[(LikerAndPoster - LikerFollowPoster)]$$

   – All the likers who have liked a post and not followed the corresponding poster and the about of those likers.

   $LikerAndAbout(Username, description) :=$

   $$\Pi_{Username/liker,description/about}\sigma_{liker=uid}[(LikerNotFollow \times User)]$$

– All the viewers and the corresponding story poster's uid.

$ViewerAndStoryPoster(viewerid, uid) :=$

$$\Pi_{viewerid,uid}[(Story \bowtie Saw)]$$

– All the viewers who have viewed a story and followed the corresponding poster.

$ViewerFollowStoryPoster(viewerid, uid) :=$

$$\Pi_{viewerid,uid}\sigma_{viewerid=follower \wedge uid=followed}[(ViewerAndStoryPoster \times Follows)]$$

– All the viewers who have viewed a story and not followed the corresponding poster.

$ViewerNotFollow(viewerid) :=$

$$\Pi_{viewerid}[(ViewerAndStoryPoster - ViewerFollowStoryPoster)]$$

– All the viewers who have viewed a story and not followed the corresponding poster and the about of those viewers.

$ViewerAndAbout(Username, description) :=$

$$\Pi_{Username/viewer,description/about}\sigma_{liker=uid}[(ViewerNotFollow \times User)]$$

– All the users who have never liked or viewed a post or story of a user that they do not follow.

$$\Pi_{Username/uid,description/about}[(User)] - ViewerAndAbout - LikerAndAbout$$

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.

– All the tags with different captions and their time in 2017.

$TagCaption(tag, caption, when) :=$

$$\Pi_{tag,caption,when}\sigma_{when.year=2017}[(Post \times hashtag)]$$

– All the tags with at least three mentioned in 2017 on at least one day.

$AtLeastThree(tag, when) :=$

$$\Pi_{T1.tag,T1.when}\sigma_{T1.tag=T2.tag=T3.tag \wedge T1.caption>T2.caption \wedge T1.caption>T3.caption \wedge T2.caption>T3.caption}$$
$$[(\rho_{T1}TagCaption) \times (\rho_{T2}TagCaption) \times (\rho_{T3}TagCaption)]$$

– Everyday in 2017.

$Everyday(when) :=$

$$\Pi_{when}\sigma_{when.year=2017}[(Post)]$$

– All tags in 2017.

$AllTag(tag) :=$

$$\Pi_{tag,when}\sigma_{when.year=2017}[(hashtag)]$$

– A table contains all tags assuming they all appear in every day of 2017 .

$AllTagEveryday(tag, when) :=$

$$\Pi_{tag,when}[(AllTag \times Everyday)]$$

– All the hashtags that have been mentioned in at least three post captions not on every day of 2017.

$NotEvery(tag) :=$

$$\Pi_{tag}[(AllTagEveryday - AtLeastThree)]$$

– Every hashtag that has been mentioned in at least three post captions on every day of 2017.

$AllTag - NotEvery$

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.

– All the reciprocal followers.

$ReciprocalFollowers(p1, p2) :=$

$$\Pi_{p1/F1.follower,p2/F2.follower}\sigma_{F1.follower=F2.followed \wedge F1.followed=F2.follower \wedge F1.follower>F2.follower}$$
$$[(\rho_{F1}Follows \times \rho_{F2}Follows)]$$

– All the followers of the first user in each pair of reciprocal followers.

$P1Followers(p1, p2, follower) :=$

$$\Pi_{p1,p2,follower}\sigma_{p1=followed}[(ReciprocalFollowers \times Followes)]$$

– All the followers of the second user in each pair of reciprocal followers.

$P2Followers(p1, p2, follower) :=$

$$\Pi_{p1,p2,follower}\sigma_{p2=followed}[(ReciprocalFollowers \times Followes)]$$

– Uncommon followers with only uid.

$UncommonFollowers(p1, p2, uid) :=$

$$\Pi_{p1,p2,uid/follower}[(P1Followers - P2Followers) \cup (P2Followers - P1Followers)]$$

– Final result, uncommon followers with p1, p2, uid, name and email.

$$\Pi_{p1,p2,uid,name,email}\sigma_{UncommonFollowers.uid=User.uid}[(UncommonFollowers \times User)]$$

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.

   – Cannot be expressed.

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.

   – All the reciprocal followers.

   $ReciprocalFollowers(p1, p2) :=$

   $$\Pi_{p1/F1.follower,p2/F2.follower}\sigma_{F1.follower=F2.followed \wedge F1.followed=F2.follower \wedge F1.follower>F2.follower}[(\rho_{F1}Follows \times \rho_{F2}Follows)]$$

   – All the pids which the first user posts in each pair of reciprocal followers.

   $P1Post(p1, p2, pid) :=$

   $$\Pi_{p1,p2,pid}\sigma_{p1=uid}[(ReciprocalFollowers \times Post)]$$

   – All the pids which the second user likes.

   $P2Like(p1, p2, pid) :=$

   $$\Pi_{p1,p2,pid}\sigma_{p2=uid}[(ReciprocalFollowers \times Likes)]$$

   – All the pids which the second user posts in each pair of reciprocal followers.

   $P2Post(p1, p2, pid) :=$

   $$\Pi_{p1,p2,pid}\sigma_{p2=uid}[(ReciprocalFollowers \times Post)]$$

   – All the pids which the first user likes.

   $P1Like(p1, p2, pid) :=$

   $$\Pi_{p1,p2,pid}\sigma_{p1=uid}[(ReciprocalFollowers \times Likes)]$$

   – All pairs of reciprocal followers who are not backscratchers.

   $NotBackscratchers(p1, p2) :=$

   $$\Pi_{p1,p2}[(P1Post - P2Like)] \cup \Pi_{p1,p2}[(P2Post - P1Like)]$$

   – All pairs of backscratchers.

   $Backscratchers(p1, p2) :=$

   $ReciprocalFollowers - NotBackscratchers$

   – All followers of P1.

   $P1Followers(follower, p1) :=$

   $$\Pi_{follower,p1}\sigma_{p1=followed}[(Follows \times Backscratchers)]$$

   – Final result, followers of at least one pair of P1 and P2.

   $$\Pi_{follower}\sigma_{p2=followed \wedge P1Followers.p1=Backscratchers.p1 \wedge P1Followers.follower=Followers.follower}[(Follows \times Backscratchers \times P1Followers)]$$

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.

   – All the posts that have at least one post after them.

   $NotMostRecentPost(pid, uid, when) :=$

   $$\Pi_{P1.pid,P1.uid,P1.when}\sigma_{P1.pid=P2.uid \wedge P1.when<P2.when}[(\rho_{P1}Post) \times (\rho_{P2}Post)]$$

   – Each user who has at least one post and his/her most recent post.

   $MostRecentPost(pid, uid, when) :=$

   $$\Pi_{pid,uid,when}[(Post)] - NotMostRecentPost$$

   – All the stories that have at least one story posted after them.

   $NotMostRecentStory(sid, uid, when) :=$

   $$\Pi_{S1.sid,S1.uid,S1.when}\sigma_{S1.sid=S2.uid \wedge S1.when<S2.when}[(\rho_{S1}Story) \times (\rho_{S2}Story)]$$

   – Each user who has at least one story posted and his/her most recent story.

   $MostRecentStory(sid, uid, when) :=$

   $$\Pi_{sid,uid,when}[(Story)] - NotMostRecentStory$$

   – Each user's uid and most recent activity time.

   $MostRecentActivity(uid, when) :=$

   $$\Pi_{P.uid,P.when}\sigma_{P.uid=S.uid \wedge P.when \geq S.when}[(\rho_P MostRecentPost) \times (\rho_S MostRecentStory)]$$
   $$+\Pi_{S.uid,S.when}\sigma_{P.uid=S.uid \wedge P.when<S.when}[(\rho_P MostRecentPost) \times (\rho_S MostRecentStory)]$$

   – Each user's uid and all their followed's uids and those followed's most recent activity time.

   $AllFollowedMostRecent(follower, followed, when) :=$

   $$\Pi_{follower,followed,when}\sigma_{followed=uid}[(Follows) \times (MostRecentActivity)]$$

   – Each user's uid and their most recently active followed's uid and time.

   $MostRecentUserOfFollower(follower, followed, when) :=$

   $$AllFollowedMostRecent -$$

$$\Pi_{follower,followed,when}\sigma_{P1.follower=P2.follower \wedge P1.when<P2.when}$$
$$[(\rho_{P1}AllFollowedMostRecent) \times (\rho_{P2}AllFollowedMostRecent)]$$

– Final result.

$$\Pi_{Username/U1.name,followedname/U2.name,when,U2.email}\sigma_{follower=U1.uid \wedge followed=U2.uid}$$
$$[(MostRecentUserOfFollower) \times \rho_{U1}(User) \times \rho_{U2}(User)]$$

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.

– All the posts with only the post time and their likers with the like time.

$LPWhen(liker, Pwhen, Lwhen) :=$

$$\Pi_{liker,Pwhen/Post.when,Lwhen/Likes.when}[(Likes \bowtie Post)]$$

– All the likers who have not liked in the same order of post.

$NotOrder(liker) :=$

$$\Pi_{liker/LPWhen1.liker}$$
$$\sigma_{LPWhen1.liker=LPWhen2.liker \wedge LPWhen1.Lwhen>LPWhen2.Lwhen \wedge LPWhen2.Pwhen<LPWhen2.Pwhen}$$
$$[(\rho_{LPWhen1}LPWhen) \times (\rho_{LPWhen2}LPWhen)]$$

– All the likers who have liked in the same order of post.

$Order(liker) :=$

$$\Pi_{liker}[(Likes)] - NotOrder$$

–Final result.

$$\Pi_{name,email}\sigma_{liker=uid}[(Order \times Users)]$$

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.

   – Cannot be expressed.

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.

– All the viewers with not the last stories.

$NotLast(viewerid, Lastsid) :=$

$$\Pi_{S1.viewerid, Lastsid/S1.sid}\sigma_{S1.viewerid=S2.viewerid \wedge S1.when<S2.when}[(\rho_{S1}Saw) \times (\rho_{S2}Saw)]$$

– All the viewers with last story.

$Last(viewerid, Lastsid) :=$

$$\Pi_{viewerid, Lastsid/sid}[(Saw)] - NotLast$$

– All the viewers with not the first stories.

$NotFirst(viewerid, Firstsid) :=$

$$\Pi_{S1.viewerid, Firstsid/S1.sid}\sigma_{S1.viewerid=S2.viewerid \wedge S1.when>S2.when}[(\rho_{S1}Saw) \times (\rho_{S2}Saw)]$$

– All the viewers with first story.

$First(viewerid, Firstsid) :=$

$$\Pi_{viewerid, Firstsid/sid}[(Saw)] - NotFirst$$

– Final result, uid with first and last stories.

$$\sigma_{First.viewerid=Last.viewerid}[(Last \times First)]$$

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.

    – All the posts and commenters that the post have at least three comments.

    $AtLeastThree(pid, commenter, when, text) :=$

    $$\Pi_{C1.pid, C1.commenter, C1.when, C1.text}\sigma_{C1.pid=C2.pid=C3.pid \wedge C1.text>C2.text \wedge C1.text>C3.text \wedge C2.text>C3.text}[(\rho_{C1}Comment) \times (\rho_{C2}Comment) \times (\rho_{C3}Comment)]$$

    – All the positive posts and commenters that have at least three comments.

    $Positive(pid, commenter, when) :=$

    $$\Pi_{pid, commenter, when}\sigma_{sentiment(text)="Positive"}[(AtLeastThree)]$$

    – All the negative posts and commenters that have at least three comments.

$Negative(pid, commenter, when) :=$

$$\Pi_{pid,commenter,when}\sigma_{sentiment(text)=\text{"}Negative\text{"}}[(AtLeastThree)]$$

– At least one negative comment happened before positive comment.

$NegativeLeft(pid) :=$

$$\Pi_{P.pid}\sigma_{P.pid=N.pid \wedge P.when>N.when}[(\rho_P Positive) \times (\rho_N Negative)]$$

– At least one positive comment happened before negative comment.

$PositiveLeft(pid) :=$

$$\Pi_{P.pid}\sigma_{P.pid=N.pid \wedge P.when<N.when}[(\rho_P Positive) \times (\rho_N Negative)]$$

– All the shifted posts.

$ShiftedPost(pid) :=$

$$\Pi_{pid}[(AtLeastThree)] - (NegativeLeft \cap PositiveLeft)$$

– Final result.

$$\Pi_{uid,commenter,Comment.when,sentiment(text)}[(Post) \bowtie (Comment) \bowtie (ShiftedPost)]$$

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.)

   $$\sigma_{Comment.when<Post.when}[(Comment) \times (Post)] = \emptyset$$

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

   $$\sigma_{S1.uid=S2.uid \wedge S1.sid \neq S2.sid \wedge S1.current=True \wedge S2.current=True}[(\rho_{S1} Story) \times (\rho_{S2} Story)] = \emptyset$$

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

   $$(\Pi_{pid}[(Post)] - \Pi_{pid}[(PIncludes)]) \cup (\Pi_{pid}[(Story)] - \Pi_{pid}[(SIncludes)]) = \emptyset$$