University of Toronto
CSC343, Fall 2017

Assignment 1:

Yifan Fang 1002563825
Zhaoning Kong 1004654288

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".

   – The pair of Uids of users who sent a like and who received this like.

   $LikerAndReceiver(likerUid, receiverUid) :=$

   $$\Pi_{Likes.liker,Post.uid}(Likes \bowtie_{Likes.pid=Post.pid} Post)$$

   – The pair of Uids selected from the relation above when the liker followed the receiver.

   $LikesFromFollower(likerUid, receiverUid) :=$

   $$\Pi_{likerUid,receiverUid}(LikerAndReceiver \bowtie_{\substack{LikerAndReceiver.likerUid=Follows.follower \wedge \\ LikerAndReceiver.receiverUid=Follows.followed}} Follows)$$

   – UID of users who liked someone he/she didn't follow.

   $UidLikedUnfollowed(uid) := \Pi_{likerUid}(LikerAndReceiver - LikesFromFollower)$

   – UID of users who never liked someone he/she didn't follow.

   $UidNotLikedUnfollowed(uid) := (\Pi_{uid}Users) - UidLikedUnfollowed$

   – The pair of Uids of users who view a story and who post this story.

   $ViewerAndPoster(viewerUid, posterUid) :=$

   $$\Pi_{Saw.viewerid,Story.uid}(Saw \bowtie_{Saw.sid=Story.sid} Story)$$

   – The pair of Uids selected from the relation above when the viewer followed the poster.

   $ViewsFromFollower(viewerUid) :=$

   $$\Pi_{viewerUid,posterUid}\sigma_{\substack{ViewerAndPoster.viewerUid=Follows.follower \wedge \\ ViewerAndPoster.posterUid=Follows.followed}}(ViewerAndPoster \bowtie Follows)$$

   – UID of users who viewed story of someone he/she didn't follow.

   $UidViewedUnfollowed(uid) := \Pi_{viewerUid}(ViewerAndPoster - ViewsFromFollower)$

   – UID of users who never viewed story of someone he/she didn't follow.

   $UidNotViewedUnfollowed(uid) := (\Pi_{uid}Users) - UidViewedUnfollowed$

   – Uid and "About" Info of such users.

   $Answer(username, description) :=$

$$\Pi_{User.uid,User.about}((UidNotLikedUnfollowed \cup UidNotViewedUnfollowed) \bowtie User)$$

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.

– Hashtags appeared on specific dates of 2017.

$TagWithDate(tag, when) := \Pi_{tag,when}\sigma_{when.year=2017}(Hashtag \bowtie Post)$

– Hashtags appeared 3+ times on specific dates of 2017.

$$TagAtLeastThreeOneDay(tag, when) := \Pi_{tag,when}\sigma_{\substack{H1.date=H2.date \wedge \\ H2.date=H3.date \wedge \\ H1.pid<H2.pid \wedge \\ H2.pid<H3.pid}}(\rho_{H1}TagWithDate \times \rho_{H1}TagWithDate \times \rho_{H1}TagWithDate)$$

–Hashtags appeared 1 or 2 times on some date of 2017.

$TagLessThanThreeOneDay(tag, when) := TagWithDate - TagAtLeastThreeOneDay$

– Hashtag mentioned in at least three post captions on every day of 2017.

$Answer(tag) := (\Pi_{tag}(TagWithDate)) - (\Pi_{tag}(TagLessThanThreeOneDay))$

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.

– Reciprocal users without order.

$$Reciprocal(userA, userB) := \Pi_{F1.follower,F1.followed}\sigma_{\substack{F1.follower=F2.followed \wedge \\ F1.followed=F2.follower \wedge \\ F1.follower<F1.followed}}(\rho_{F1}Follower \times \rho_{F2}Follower)$$

– For each pair of reciprocal users, a third user follows one of them or both.

$$FollowOneOrBoth(userA, userB, userC) := \Pi_{R.follower,R.followed,F1.follower}\sigma_{\substack{F1.follower=F2.follower \wedge \\ (F1.followed=R.userA \vee \\ F2.followed=R.userB) \wedge \\ F1.follower \neq R.userB \wedge \\ F2.follower \neq R.userA}}(\rho_{R}Reciprocal \times \rho_{F1}Follows \times \rho_{F2}Follows)$$

– For each pair of reciprocal users, a third user follows both of them.

$$FollowBoth(userA, userB, userC) := \Pi_{R.follower,R.followed,F1.follower}\sigma_{\substack{F1.follower=F2.follower \wedge \\ F1.followed=R.userA \wedge \\ F2.followed=R.userB \wedge \\ F1.follower \neq R.userB \wedge \\ F2.follower \neq R.userA}}(\rho_{R}Reciprocal \times \rho_{F1}Follows \times \rho_{F2}Follows)$$

– For each pair of reciprocal users, a third user follow one of them, but not both.

$$Answer(recA, recB, uncommon, name, email) :=$$
$$\Pi_{userA,userB,userC,name,email}((FollowOneOrBoth - FollowBoth) \bowtie_{userc=uid} User)$$

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.

   – Relation including pid and uid of poster and uid of liker.

   $$PostWithLikerUid(pid, posterUid, likerUid) :=$$
   $$\Pi_{Post.pid,Post.uid,Likes.liker}(Post \bowtie_{Post.pid=Likes.pid} Likes)$$

   – Relation including pid and uid of poster and uid of all followers of this poster.

   $$PostWithFollowerUid(pid, posterUid, followerUid) :=$$
   $$\Pi_{pid,posterUid,follower}(PostWithLikerUid \bowtie_{posterUid=followed} Follows)$$

   – uid with all of the followers' uid who do not like all his/her posts

   $$UidWithNotQualifiedFollowerUid(posterUid, followerUid) :=$$
   $$\Pi_{posterUid,followerUid}(PostWithFollowerUid - \rho_{P(pid,posterUid,followerUid)} PostWithLikerUid)$$

   – Relation including uid and all his/her followers' uid.

   $$AllFollowed(uid, follower) :=$$
   $$\Pi_{User.uid,Follows.follower}(User \bowtie_{User.uid=Follows.followed} Follows)$$

   – uid with all of the followers' uid who like all his/her posts

   $$UidWithQualifiedFollowerUid(uid, followerUid) :=$$
   $$AllFollowed - UidWithNotQualifiedFollowerUid$$

   – Self join the relation above, the result is the "Backscratchers".

   $$Backscratchers(buid1, buid2) :=$$
   $$\Pi_{U1.uid,U2.uid} \sigma_{U1.uid=U2.followerUid \wedge U1.followerUid=U2.uid \wedge U1.uid>U2.uid}$$
   $$(\rho_{U1} UidWithQualifiedFollowerUid \times \rho_{U2} UidWithQualifiedFollowerUid)$$

   – uid of all users who follow some pair of backscratcher users.

   $$Answer(uid) :=$$
   $$\Pi_{A1.follower} \sigma_{\substack{A1.follower=A2.follower \wedge \\ A1.uid=B.buid1 \wedge \\ A2.uid=B.buid2}} (\rho_B Backscratchers \times \rho_{A1} AllFollowed \times \rho_{A2} AllFollowed)$$

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.

   – uid of user and time of his/her all posts and stories.

   $UidAndTime(uid, when) := \Pi_{uid,when} Post \cup \Pi_{uid,when} Story$

   – All posts with uid of the poster, and uid of all followers of that poster.

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

   $$\Pi_{F.follower,F.followed,U.when} \sigma_{F.followed=U.uid}(\rho_F Follows \bowtie \rho_U UidAndTime)$$

   – Time of posts that are not most recent for each pair of users.

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

   $$\Pi_{C1.follower,C2.followed,C2.date} \sigma_{\substack{C1.follower=C2.follower \wedge \\ C1.when>C2.when}}(\rho_{C1} C \times \rho_{C2} C)$$

   – Most recent user(s) for each user, and date of most-recent activity.

   $MostRecentUser(follower, followed, when) := C - NotMostRecentPosts$

   – Most recent user(s) for each user, their name, email and date of most-recent activity.

   $Answer(follower, mostRecentUid, name, email, when) :=$

   $$\Pi_{U1.name,M.followed,U2.name,U2.email,M.when} \sigma_{M.follower=U1.uid \wedge M.followed=U2.uid}$$
   $$(\rho_M MostRecentUser \times \rho_{U1} User \times \rho_{U2} User)$$

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.

   – A relation representing user liker liked post pid posted at posttime at likestime.

   $Likespost(liker, pid, likestime, posttime) :=$

   $$\Pi_{Likes.liker,Post.pid,Likes.when,Post.when}(Likes \bowtie_{Likes.pid=Posts.pid} Post)$$

   – uid of users who didn't like posts in order they were posted.

   $$uidNotOrder(uid) := \Pi_{L1.uid} \sigma_{\substack{L1.uid=L2.uid \wedge \\ L1.likestime>L2.likestime \wedge \\ L1.posttime<L2.posttime}} (\rho_{L1} Likespost \times \rho_{L2} Likespost)$$

– uid of users who always liked posts in order they were posted.
$uidInOrder(uid) := (\Pi_{uid} User) - uidNotOrder.$

– Name and email of such user.
$Answer(name, email) := \Pi_{User.name, User.email}(uidInOrder \bowtie User)$

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.

   – For each user, the sid of stories except for the last he/she have seen.
   $NotLast(viewerid, sid) := \Pi_{S1.viewerid, S2.sid} \sigma_{\substack{S1.viewerid=S2.viewerid \wedge \\ S1.when>S2.when}} (\rho_{S1} Saw \times \rho_{S2} Saw)$

   – For each user, the sid of last stories he/she have seen.
   $Last(viewerid, sid) := (\Pi_{viewerid, sid} Saw) - NotLast;$

   – For each user, the sid of stories except for the first he/she have seen.
   $NotFirst(viewerid, sid) := \Pi_{S1.viewerid, S2.sid} \sigma_{\substack{S1.viewerid=S2.viewerid \wedge \\ S1.when<S2.when}} (\rho_{S1} Saw \times \rho_{S2} Saw)$

   – For each user, the sid of first stories he/she have seen.
   $First(viewerid, sid) := (\Pi_{viewerid, sid} Saw) - NotFirst;$

   – For each user, his/her uid and sid of first and last stories they have seen.
   $Answer(viewerid, sid) := First \cup Last.$

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.

– A relation including pid, uid from who comments, and the text and time of the comment.

$$AllCommentsWithPid(pid, commenterID, text, when) :=$$

$$\Pi_{Post.pid,C.commenter,C.text,C.when}(\rho_C Comment \bowtie_{C.pid=Post.pid} Post)$$

– A relation selected from above where the comments are not the earliest positive comments for each post.

$$NotEarliestPositive(pid, commenterID, when) :=$$

$$\Pi_{A2.pid,A2.commenterID,A2.when} \sigma_{\substack{sentiment(A1.text)="positive" \wedge \\ sentiment(A2.text)="positive" \wedge \\ A1.pid=A2.pid \wedge \\ A1.when<A2.when}} (\rho_{A1} AllCommentsWithPid \times \rho_{A2} AllCommentsWithPid)$$

– The earliest positive comment for each post.

$$EarliestPositive(pid, commenterID, when) :=$$

$$(\Pi_{pid,commenterID,when} \sigma_{sentiment(text)="positive"} AllCommentsWithPid) - NotEarliestPositive$$

– A relation selected from the first step where the comments are not the latest negative comments for each post.

$$NotLatestNegative(pid, commenterID, when) :=$$

$$\Pi_{A2.pid,A2.commenterID,A2.when} \sigma_{\substack{sentiment(A1.text)="negative" \wedge \\ sentiment(A2.text)="negative" \wedge \\ A1.pid=A2.pid \wedge \\ A1.when>A2.when}} (\rho_{A1} AllCommentsWithPid \times \rho_{A2} AllCommentsWithPid)$$

– The latest negative comment for each post.

$$LatestNegative(pid, commenterID, when) :=$$

$$(\Pi_{pid,commenterID,when} \sigma_{sentiment(text)="negative"} AllCommentsWithPid) - NotLatestNegative$$

– pid of all post having a sentiment shift from negative to positive.

$$AllFromNegativeToPositive(pid) :=$$

$$\Pi_{EP.pid} \sigma_{EP.pid=LN.pid \wedge EP.when>LN.when} (\rho_{EP} EarliestPositive \times \rho_{LN} LatestNegative)$$

– A relation selected from the first step where the comments are not the earliest negative comments for each post.

$$NotEarliestNegative(pid, commenterID, when) :=$$

$$\Pi_{A2.pid,A2.commenterID,A2.when}$$

$$\sigma_{\substack{sentiment(A1.text)="negative"\wedge \\ sentiment(A2.text)="negative"\wedge \\ A1.pid=A2.pid\wedge \\ A1.when<A2.when}}(\rho_{A1}AllCommentsWithPid \times \rho_{A2}AllCommentsWithPid)$$

– The earliest negative comment for each post.

$EarliestNegativepid, commenterID, when) :=$

$$(\Pi_{pid,commenterID,when}\sigma_{sentiment(text)="negative"}AllCommentsWithPid) - NotEarliestNegative$$

– A relation selected from the first step where the comments are not the latest positive comments for each post.

$NotLatestPositive(pid, commenterID, when) :=$

$$\Pi_{A2.pid,A2.commenterID,A2.when}$$
$$\sigma_{\substack{sentiment(A1.text)="positive"\wedge \\ sentiment(A2.text)="positive"\wedge \\ A1.pid=A2.pid\wedge \\ A1.when>A2.when}}(\rho_{A1}AllCommentsWithPid \times \rho_{A2}AllCommentsWithPid)$$

– The latest positive comment for each post.

$LatestNegative(pid, commenterID, when) :=$

$$(\Pi_{pid,commenterID,when}\sigma_{sentiment(text)="positive"}AllCommentsWithPid) - NotLatestPositive$$

– pid of all post having a sentiment shift from positive to negative.

$AllFromPositiveToNegative(pid) :=$

$$\Pi_{EN.pid}\sigma_{EN.pid=LP.pid\wedge EN.when>LP.when}(\rho_{EN}EarliestNegative \times \rho_{LP}LatestPositive)$$

– All the posts which have a sentiment shift, with its uid who owns it, each comment on the post, the commenters id, the date of their comment and its sentiment.

$Answer(pid, uid, commenterID, when, sentiment) :=$

$$\Pi_{Post.pid,Post.uid,Comment.commenter,Comment.when,sentiment(text)}$$
$$[(AllFromNegativeToPositive \cup AllFromPositiveToNegative) \bowtie$$
$$(Comment \bowtie_{Comment.pid=Post.pid} Post)]$$

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.)

$$\sigma_{Post.when>Comment.when}(Post \bowtie_{Post.pid=Comment.pid} Comment) = \emptyset$$

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

$$\sigma_{\substack{S1.uid=S2.uid \wedge S1.sid>S2.sid \wedge \\ S1.current="yes" \wedge S2.current="yes"}}(\rho_{S1}Story \times \rho_{S2}Story) = \emptyset$$

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

   – All pid of posts that does not have a photo or video.
   $UnqualifiedPost(ID) := (\Pi_{pid}Posts) - (\Pi_{pid}PIncludes)$

   – All sid of stories that does not have a photo or video.
   $UnqualifiedStory(ID) := (\Pi_{sid}Story) - (\Pi_{pid}SIncludes)$

   $UnqualifiedPost \cup UnqualifiedStory = \emptyset$