CSC343, Fall 2017 A1

Peiqi Wang (1001132561), Xiaoye Yuan (1002342084)

October 11, 2017

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".

   – A liked B's post
   $AlikedB(A, B) :=$
   $$\rho_{AlikesB(A,B)}\Pi_{liker,uid}(Likes \bowtie Post)$$
   – A saw B's story
   $AsawB(A, B) :=$
   $$\rho_{AsawB(A,B)}\Pi_{viewerid,uid}(Saw \bowtie Story)$$
   – A did not follow B
   $ANotFollowB(A, B) :=$
   $$\big(\rho_{UserCombination(follower,followed)}\Pi_{U1.uid,U2.uid}(\rho_{U1}User \times \rho_{U2}User)\big)$$
   $$- \Pi_{follower,followed}Follows$$

   – User who have liked a post or saw a story of a user they do not follow
   $TheCuriousOnes(uid) :=$
   $$\Pi_{U.uid}\sigma_{\substack{U.uid=NF.A\wedge \\ ((U.uid=L.A\wedge L.B=NF.B)\vee(U.uid=S.A\wedge S.B=NF.B))}}$$
   $$(\rho_U User \times \rho_{NF}ANotFollowB \times \rho_L ALikesB \times \rho_S ASawB)$$

   – user who have not liked a post or saw a story of user they do not follow
   $TheUncuriousOnes(uid) :=$
   $$(\Pi_{uid}User) - TheCuriousOnes$$
   $Solution(username, description) :=$
   $$\rho_{Solution(username,description)}\Pi_{uid,about}(TheUncuriousOnes \bowtie User)$$

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.

– Tags in 2017
$Post2017(pid, uid, when, location, caption) :=$

$$\sigma_{when.year=2017} Post$$

– Combine tag and time info in same place
$PostAndTag(pid, tag, when) :=$

$$\Pi_{pid,tag,when}(Post2017 \bowtie Hashtag)$$

– Hashtags mentioned in at least 3 post captions
$PossiblyPopularTags(tag, when) :=$

$$\Pi_{tag,when} \sigma_{\substack{T1.when=T2.when=T3.when \\ \wedge\, T1.tag=T2.tag=T3.tag \\ \wedge\, (T1.pid!=T2.pid) \wedge (T1.pid!=T3.pid) \wedge (T2.pid!=T3.pid)}}$$

$$(\rho_{T1} PostAndTag \times \rho_{T2} PostAndTag \times \rho_{T2} PostAndTag)$$

– Expected tag-when pair
$PopularTagsChecklist(tag, when) :=$

$$\Pi_{tag} PossiblyPopularTags \times \Pi_{when} Post$$

– Tags that fails to be popular for >= 1 days
$NotConsistentlyPopularTags(tag, when) :=$

$$PopularTagsChecklist - PossiblyPopularTags$$

– Tags that is popular for all days in a year
$Solution(tag) :=$

$$\Pi_{tag} PossiblyPopularTags - \Pi_{tag} NotConsistentlyPopularTags$$

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.

– Create a table containing all reciprocal followers without duplication
$ReciprocalFollowers(follower, followed) :=$

$$\rho_{ReciprocalFollowers(follower,followed)}$$

$$\Pi_{F1.follower,F2.follower} \left(\sigma_{\substack{F1.follower=F2.followed \\ \wedge F2.follwer=F1.followed \\ \wedge F1.follower<F1.followed}} (\rho_{F1} Follows \times \rho_{F2} Follows) \right)$$

– Create a table containing all followers who follows at least one of the reciprocal followers A and B
$AllFollowers(follower, followed) :=$

$$\rho_{AllFollowers(follower,followed)}\Pi_{Follows.follower,Follows.followed}\left(\sigma_{\substack{Follows.followed=ReciprocalFollowers.follower\\ \vee Follows.followed=ReciprocalFollowers.followed}}(Follows \times ReciprocalFollowers)\right)$$

– Create a table containing all followers who follows both reciprocal followers
$CommonFollowers(follower) :=$

$$\rho_{CommonFollowers(follower)}$$

$$\Pi_{f_3.follower}\left(\sigma_{\substack{f_3.followed=ReciprocalFollowers.follower\\ \wedge f_4.followed=ReciprocalFollowers.followed\\ \wedge f_3.follower=f_4.follower}}(\rho_{f_3}Follows \times \rho_{f_4}Follows \times ReciprocalFollowers)\right)$$

–Create a table to store the identifier for all uncommon followers
$UncommonFollowersIndentifier(follower) :=$

$$\Pi_{follower}AllFollowers - CommonFollowers$$

–Final result
$UncommonFollower(reciprocalfollower, uncommonfollower, name, email) :=$

$$\rho_{UncommonFollower(reciprocalfollower,uncommonfollower,name,email)}$$
$$\Pi_{f_5.followed,f_5.follower,name,email}$$
$$\sigma_{\substack{f_5.follower=UncommonFollowersIndentifier.follower\\ \wedge(f_5.followed=ReciprocalFollowers.follower \vee f_5.followed=ReciprocalFollowers.followed)\\ \wedge f_5.follower=User.uid}}$$
$$(\rho_{f_5}Follows \times ReciprocalFollowers \times UncommonFollowersIndentifier \times User)$$

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.

   – A and B follows each other, appear twice in table
   $FollowEachOther(A, B) :=$

$$\rho_{FollowEachOther(A,B)}\Pi_{F1.follower,F1.followed}\sigma_{\substack{F1.follower=F2.followed\\ F1.followed=F2.follower}}(\rho_{F1}Follows \times \rho_{F2}Follows)$$

– Relation containing B's posts for which the A liked
$ALikedBPost(A, B, pid) :=$

$$\rho_{ALikedBPost(A,B,pid)} \left(\Pi_{F.A,F.B,L.pid} \sigma_{\substack{F.A=L.liker \\ F.B=P.uid \\ L.pid=P.pid}} (\rho_F FollowEachOther \times \rho_L Likes \times \rho_P Post) \right)$$

– Expected relation where A liked every post made by B given that A and B follows each other
$ALikedBEveryPost(A, B, pid) :=$

$$\rho_{ALikedBEveryPost(A,B,pid)} \Pi_{F.A,F.B,P.pid} (\rho_F FollowEachOther \bowtie_{F.B=P.uid} \rho_P Post)$$

– Relation where A did not liked every post made by B
$DidNotLikedEveryPost(A, B, pid) :=$

$$ALikedBEveryPost - ALikedBPost$$

– Observed relation where A liked every post made by B
$AIsBackScratcherOfB(A, B) :=$

$$\Pi_{A,B} ALikedBPost - \Pi_{A,B} DidNotLikedEveryPost$$

– Observed relation where A liked every post made by B. and vice versa
$ABAreBackScratchers(A, B) :=$

$$\rho_{ABAreBackScratchers(A,B)} \Pi_{T1.A,T1.B}$$

$$\rho_{T1} AIsBackScratcherOfB \bowtie_{T1.A=T2.B \wedge T1.B=T2.A} \rho_{T2} AIsBackScratcherOfB$$

– Solution
$Solution(uid) :=$

$$\rho_{Solution(uid)} \Pi_A ABAreBackScratchers$$

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.

   – User's not most recent posts
   $NotMostRecentPost(uid, when) :=$

$$\rho_{NotMostRecentPost(uid,when)} \Pi_{P1.uid,P1.when}$$
$$\sigma_{P1.when<P2.when} (\rho_{P1} Post \times \rho_{P2} Post)$$

   – User's most recent posts
   $MostRecentPost(uid, when) :=$

$$\Pi_{uid,when} Post - NotMostRecentPost$$

– User's not most recent stories
$NotMostRecentStory(uid, when) :=$

$$\rho_{NotMostRecentStory(uid,when)}\Pi_{S1.uid,S1.when} \sigma_{S1.when<S2.when}\,(\rho_{S1}Story \times \rho_{S2}Story)$$

– User's most recent stories
$MostRecentStory(uid, when) :=$

$$\Pi_{uid,when}Story - NotMostRecentStory$$

– User's most recent post and most recent story in one table
$MostRecentCombined(uid, when) :=$

$$MostRecentStory \bigcup MostRecentPost$$

– User's most recent activity, one for each user, unless there is a tie
$MostRecentActivity(uid, when) :=$

$MostRecentCombined\ -$

$$\rho_{NotMostRecentActivity(uid,when)}\Pi_{C1.uid,C1.when} \sigma_{C1.when<C2.when}(\rho_{C1}MostRecentCombined \times \rho_{C2}MostRecentCombined)$$

– Relation containing users, and all users they follow with their respective most recent activity time
$FollowActivity(A, B, when) :=$

$$\rho_{FollowActivity(A,B,when)}\Pi_{F.follower,F.followed,M.when} (\rho_F Follows \bowtie_{F.followed=M.uid} \rho_M MostRecentActivity)$$

– Relation containing users, and only the most recently active users they follow
$MostRecentlyActiveFollow(A, B, when) :=$

$FollowActivity\ -$

$$\rho_{NotRecentlyActiveFollow(A,B,when)}\Pi_{F1.A,F1.B,F1.when} \sigma_{F1.when<F2.when}(\rho_{F1}FollowActivity \bowtie_{F1.A=F2.A} \rho_{F2}FollowActivity)$$

– Add name and email to previous relation
$Solution(name, followername, followeremail, followerwhen) :=$

$$\rho_{Solution(name,followername,followeremail,followerwhen)} \Pi_{U1.name,U2.name,U2.email,M.when} (\rho_M MostRecentlyActiveFollow \bowtie_{M.A=U1.uid} \rho_{U1} User \bowtie_{M.B=U2.uid} \rho_{U2} User)$$

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.

– Relation that augment $Likes$ to include when post is posted
$LikeWhen(uid, pid, postwhen, likewhen) :=$

$$\rho_{LikeWhen(uid,pid,postwhen,likewhen)}\Pi_{L.uid,L.pid,P.when,L.when}(\rho_P Post \bowtie \rho_L Likes)$$

– Relation where user does not follow the ordering specified above
$NotOrderedUser(uid) :=$

$\rho_{NotOrderedUser(uid)}\Pi_{L1.uid}$
$\sigma_{\substack{(L1.postwhen<L2.postwhen \wedge L1.likewhen \geq L2.likewhen) \\ \vee(L1.postwhen>L2.postwhen \wedge L1.likewhen \leq L1.likewhen)}} (\rho_{L1} LikeWhen \bowtie_{L1.uid=L2.uid} \rho_{L2} LikeWhen)$

– Relation where user does follow the ordering specified above
$OrderedUser(uid) :=$

$$(\rho_{all(uid)}\Pi_{liker}Likes) - NotOrderedUser$$

– Solution
$Solution(name, email) :=$

$$\sigma_{name,email}(OrderedUser \bowtie User)$$

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.

– Create a table to record viewers and stories they viewed that weren't the last stories they viewed
$NotLast(viewerid, sid) :=$

$$\rho_{NotLast(viewerid,sid)}\Pi_{s_1.viewerid,s_1.sid}\sigma_{\substack{s_1.viewerid=s_2.viewerid \\ s_1.sid \neq s_2.sid \\ s_1.when<s_2.when}}(\rho_{s_1} Saw \times \rho_{s_2} Saw)$$

– Create a table to record viewers and stories they viewed that were the last stories they viewed
$Last(viewerid, sid) :=$

$$\Pi_{viewerid,sid}Saw - NotLast$$

– Create a table to record viewers and stories they viewed that weren't the first stories they viewed
$NotFirst(viewerid, sid) :=$

$$\rho_{NotFirst(viewerid,sid)}\Pi_{s_1.viewerid,s_1.sid}\sigma_{\substack{s_1.viewerid=s_2.viewerid\\ s_1.sid\neq s_2.sid\\ s_1.when>s_2.when}}(\rho_{s_1}Saw \times \rho_{s_2}Saw)$$

– Create a table to record viewers and stories they viewed that were the first stories they viewed
$First(viewerid, sid) :=$

$$\Pi_{viewerid,sid}Saw - NotFirst$$

– Create a table to record each viewer's first and last viewed stories
$FirstAndLast(viewerid, firstsid, lastsid) :=$

$$\rho_{FirstAndLast(viewerid,firstsid,lastsid)}\Pi_{First.viewerid,First.sid,Last.sid}$$

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

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.

    – Posts with at least 3 different comment. Comments for a single post are uniquely identified by commenter and when. So a post has at least three unique comments if pairwise inequality operation of 3 cross product of the *Comment* yields true conjunctively
    $HotPostPid(pid) :=$

$$\rho_{HotPost(pid)}\Pi_{C1.pid}$$

$$\left(\sigma_{\substack{(C1.pid=C2.pid=C3.pid)\\ \wedge(C1.commenter\neq C2.commenter \vee C1.when!=C2.when)\\ \wedge(C1.commenter\neq C3.commenter \vee C1.when!=C3.when)\\ \wedge(C2.commenter\neq C3.commenter \vee C2.when!=C3.when)}}(\rho_{C1}Comment \times \rho_{C2}Comment \times \rho_{C3}Comment)\right)$$

– As per explained by prof Horton, comments of a post that only has positive or only has negative sentiment does not count as sentiment shift. So here we filter *HotPostPid*

to exclude such cases
$HostPostPidFiltered(pid) :=$

$\rho_{HostPostPidFiltered(pid)}\Pi_{pid}$
$\sigma_{sentiment(H1.text)\neq sentiment(H2.text)}(\rho_{H1}(HotPostPid \bowtie Comment) \times \rho_{H2}(HotPostPid \bowtie Comment))$

– Filters *Comment* such that only comments belonging to a post with at least 3 comments exist and there should be a sentiment change
$HotPostComments(pid, commenter, when, text) :=$

$$HostPostPidFiltered \bowtie Comment$$

– Posts with at least 3 different comments and for which there is not a sentiment change over time. A three tuple of *HotPostComments* does not have a sentiment change iff there is a sequence of "positive"-"negative"-"positive" or "negative"-"positive"-"negative" of comments when ordered by time
$NotSentimentHotPost(pid) :=$

$$\rho_{NotSentimentHotPost(pid)}\Pi_{C1.pid}\ \sigma_{\substack{(C1.pid=C2.pid=C3.pid)\\ \wedge(C1.when<C2.when<C3.when)\\ \wedge((sentiment(C1.text)=``positive'' \wedge sentiment(C2.text)=``negative'' \wedge sentiment(C3.text)=``positive'')\vee\\ (sentiment(C1.text)=``negative'' \wedge sentiment(C2.text)=``positive'' \wedge sentiment(C3.text)=``negative''))}}$$
$$(\rho_{C1}HotPostComments \times \rho_{C2}HotPostComments \times \rho_{C3}HotPostComments)$$

– Posts with at least 3 different comments and there is a sentiment shift
$SentimentHotPost(pid) :=$

$$HostPostPidFiltered - NotSentimentHotPost$$

– Adds owner info to above relation
$Solution(ownerid, commenter, commentwhen, commentsentiment) :=$

$$\rho_{Solution(ownerid,commenter,commentwhen,commentsentiment)}$$
$$\Pi_{P.uid,C.uid,C.when,sentiment(C.text)}(\rho_T SentimentHotPost \bowtie \rho_P Post \bowtie \rho_C Comment)$$

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.)

$$(\sigma_{P.when>C.when}(\rho_C Comment \bowtie \rho_P Post)) = \emptyset$$

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

$$\left(\sigma_{\substack{S1.current=\text{“}yes\text{''} \wedge S2.current=\text{“}yes\text{''} \\ S1.sid<S2.sid}}(\rho_{S1} Story \bowtie_{S1.uid=S2.uid} \rho_{S1} Story)\right) = \emptyset$$

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

$$\big(\rho_{PostWithNoIncludes(id)}(\Pi_{pid} Post - \Pi_{pid} PIncludes)\big)$$
$$\bigcup \big(\rho_{StoryWithNoIncludes(id)}(\Pi_{sid} Story - \Pi_{sid} SIncludes)\big) = \emptyset$$