University of Toronto
CSC343, Fall 2017

Assignment 1: Sample Solutions

Note that there are multiple correct answers to all of these questions.

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".

   SOLUTION:

   – all possible pairs of users

   $AllUserPairs(user1, user2) := \Pi_{U1.uid, U2.uid}\sigma_{U1.uid>U2.uid}[\rho_{U1}Users \times \rho_{U2}Users]$

   – pairs where user1 does not follow user2

   $NotFollowed(user1, user2) := AllUserPairs - \rho(user1, user2)(\Pi_{Follower,Followed}Follows)$

   – users who liked post of someone they don't follow

   $LikedStrangerPost(user) := \Pi_{user1}\sigma_{user2=uid}NotFollowed \times (Likes \bowtie Post)$

   – users who viewed story of someone they do not follow

   $ViewedStrangerStory(user) := \Pi_{user1}$
   $\sigma_{NotFollowed.user1=Saw.viewerid \wedge NotFollowed.user2=Story.uid \wedge Saw.sid=Story.sid}(Saw \times Story \times NotFollowed)$

   – users who did any of the 2 things

   $AnyStranger(uid) := LikedStrangerPost \cup ViewedStrangerStory$

   – users who did NOT do either of these two things

   $Candidates(uid) := \Pi_{uid}Users - AnyStranger$

   – final (join and rename to get fields we need)

   $Final(name, description) := \rho(name, description)(\Pi_{name,about}(Candidates \bowtie Users)$

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.

   SOLUTION:

   – find posts in 2017, keep day and pid

   $2017posts(day, pid) := \Pi_{when.day,pid}\sigma_{when.year=\text{"2017"}}Post$

   – find all days in 2017

   $2017days(day) := \Pi_{when.day}\sigma_{when.year=\text{"2017"}}Post$

   – find hashtags with their posts from 2017

   $HT2017(day, tag, pid) := 2017posts \bowtie Hashtag$

– find those mentioned on 3 times on same day
$ThreeMentions(day, tag) :=$

$$\Pi_{HT1.day,HT1.tag}$$
$$\sigma_{(HT1.tag=HT2.tag=HT3.tag)\wedge(HT1.day=HT2.day=HT3.day)\wedge(HT1.pid>HT2.pid>HT3.pid)}$$
$$[(\rho_{HT1}HT2017)\times(\rho_{HT2}HT2017)\times(\rho_{HT3}HT2017)]$$

– make table of all days in 2017 and all tags
$AllDaysTags(day, tag) := 2017days \times \Pi_{tag}Hashtag$

– find day,tag combos that did not have 3 mentions
$NotMentionedThrice(day, tag) := AllDaysTags - ThreeMentions$

– set difference on tags only to find final solution
$Final(tag) := \Pi_{tag}AllDaysTags - \Pi_{tag}NotMentionedThrice$

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.

   Solution:

   $ReciprocalFollowers(user1, user2) :=$
   $$\Pi_{F1.follower,F2.follower}\sigma_{\substack{F1.follower=F2.followed\wedge\\ F1.followed=F2.follower\wedge\\ F1.follower<F2.follower}}[(\rho_{F1}Follows)\times(\rho_{F2}Follows)]$$

   – find sets of followers of each of the two users
   $FollowsUser1(user1, user2, follower) := \Pi_{RF.user1,RF.user2,F.follower}\sigma_{RF.user1=F.followed}$
   $$[(\rho_{RF}ReciprocalFollowers)\times(\rho_{F}Follows)]$$

   $FollowsUser2(user1, user2, follower) := \Pi_{RF.user1,RF.user2,F.follower}\sigma_{RF.user2=F.followed}$
   $$[(\rho_{RF}ReciprocalFollowers)\times(\rho_{F}Follows)]$$

   $FollowsBoth(user1, user2, follower) := FollowsUser1 \cap FollowsUser2$

   $FollowsEither(user1, user2, follower) := FollowsUser1 \cup FollowsUser2$

   $FollowsOne(user1, user2, follower) := FollowsEither - FollowsBoth$

   $Solution(user1, user2, name, email) :=$
   $$\Pi_{user1,user2,name,email}\sigma_{follower=uid}[FollowsOne \times User]$$

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.

   Solution: Not possible with relational algebra alone.

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.

   SOLUTION:

   – find reciprocal follower pairs

   $ReciprocalFollowers(user1, user2) := \Pi_{F1.follower, F2.follower}$

   $\sigma_{F1.follower=F2.followed \wedge F1.followed=F2.follower \wedge F1.follower<F2.follower}$

   $[\rho_{F1} Follows \times \rho_{F2} Follows]$

   – find all posts by user1

   $AllPostsU1(pid, user1, user2) := \Pi_{pid,user1,user2}\sigma_{uid=user1}[ReciprocalFollowers \times Post]$

   – find posts by user1 liked by user2

   $LikedU1Posts(pid, user1, user2) := \Pi_{pid,user1,user2}\sigma_{(uid=user1)\wedge(liker=user2)\wedge(Post.pid=Likes.pid)}$

   $[ReciprocalFollowers \times Post \times Likes]$

   – posts by user1 not liked by user2

   $NotLikedU1Posts(pid, user1, user2) := AllPostsU1 - LikedPosts$

   – pairs where user1 doesn't like ALL of user2 posts

   $NotReciprocalLikerU1(user1, user2) := \Pi_{user1,user2} NotLikedU1Posts$

   – now do most of this again to find pairs where user2 doesn't like ALL of user1 posts

   – find all posts by user2

   $AllPostsU2(pid, user1, user2) := \Pi_{pid,user1,user2}\sigma_{uid=user2}[ReciprocalFollowers \times Post]$

   – find posts by user2 liked by user1

   $LikedU1Posts(pid, user1, user2) := \Pi_{pid,user1,user2}\sigma_{(uid=user2)\wedge(liker=user1)\wedge(Post.pid=Likes.pid)}$

   $[ReciprocalFollowers \times Post \times Likes]$

   – posts by user2 not liked by user1

   $NotLikedU2Posts(pid, user1, user2 := AllPostsU2 - LikedU2Posts$

   – pairs where user2 doesn't like ALL of user1 posts

   $NotReciprocalLikerU2(user1, user2) := \Pi_{user1,user2} NotLikedU2Posts$

   – Backscratchers are all Reciprocal followers once re remove the NotReciprocalLikers in both directions

   $Backscratchers(user1, user2) := ReciprocalFollowers - NotReciprocalLikerU1 - NotReciprocalLikerU2$

   – Final (join Follows with Backscratchers)

   $Final(user) := \Pi_{F1.follower}$

$\sigma_{(F1.Follower=F2.Follower)\wedge(F1.Followed=Backscratchers.user1)\wedge(F2.Followed=Backscratchers.user2)}$
$[\rho_{F1}Follows \times \rho_{F2}Follows \times Backscratchers]$

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.

   SOLUTION:

   – make table of users and dates of activity (include both posts and stories)

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

   – find last date per user (all dates - not-last-dates)

   $NotLastActivity(uid, when) := \Pi_{(AD1.uid),(AD1.when)}\sigma_{(AD1.uid)=(AD2.uid)\wedge(AD1.when)<(AD2.when)}$
   $[\rho_{AD1}ActivityDate \times \rho_{AD2}ActivityDate]$

   $LastActivity(uid, when) := ActivityDate - NotLastActivity$

   – for each follower, find all followed and last-dates

   $FollowedLastDate(follower, followed, when) := \Pi_{follower,uid,when}\sigma_{(Follows.followed)=(LastActivity.uid)}$
   $[Follows \times LastActivity]$

   – for each follower find the followed that is not most recent

   $NotMostRecent(follower, followed, when) := \Pi_{F1.follower,F1.followed,F1.when}$
   $\sigma_{F1.follower=F2.follower\wedge F1.when<F2.when}$
   $[\rho_{F1}FollowedLastDate \times \rho_{F2}FolllowedLastDate]$

   – for each follower, find the followed that is most recent

   $MostRecent(follower, followed, when) := FollowedLastDate - NotMostRecent$

   – report the required info

   $Result(followerName, followedName, email, when) := \Pi_{Follower.name,Followed.name,Followed.email,when}$
   $\sigma_{Follower.uid=MostRecent.follower\wedge Followed.uid=MostRecent.followed}$
   $[\rho_{Follower}User \times \rho_{Followed}User \times MostRecent]$

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.

SOLUTION:

– find ordered pairs of posts and likes that do not match

$LikedOldPosts(uid) := \Pi_{L1.liker}$

$$\sigma_{P1.pid=L1.pid \wedge P2.pid=L2.pid \wedge P1.when<P2.when \wedge L1.when>L2.when \wedge L1.liker=L2.liker}$$
$$[\rho_{P1} Post \times \rho_{P2} Post \times \rho_{L1} Likes \times \rho_{L2} Likes]$$

– find uid of users we need for final answer with subtraction

$NeverLikedOldPosts(uid) := \Pi_{uid} User - LikedOldPosts$

– put together the fields that we require for the final answer

$Result(name, email) := Pi_{name,email} NeverLikedOldPosts \bowtie User$

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.

   SOLUTION: Cannot be expressed by RA.

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.

   SOLUTION:

   – Views that are not last

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

   – Last Story seen for this user

   $LastStory(viewerid, last) := (\Pi_{viewerid,sid} Saw) - NotLastSaw$

   – Do the same for First (NotFirstSaw etc.)
   – For demonstration reasons, I'll do this one a slightly different way

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

   $FirstStory(viewerid, first) := \Pi_{viewerid,sid}[Saw - NotFirstSaw]$

   – Now natural join together to get all 3 columns

   $Result(viewerid, first, last) := LastStory \bowtie FirstStory$

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.

SOLUTION:

– Table with sentiment and time of each comment

$Sent(pid, sentiment, when) := \Pi_{pid, sentiment(text), when} Comment$

– Find posts that shift and then shift back again

$DoubleShiftingPosts(pid) := \Pi_{pid}$

$\sigma_{S1.pid=S2.pid=S3.pid \wedge S1.when<S2.when<S3.when \wedge S1.sentiment=S3.sentiment \wedge S1.sentiment!=S2.sentiment}$
$[\rho_{S1} Sent \times \rho_{S2} Sent \times \rho_{S3} Sent]$

– Find all posts that have at least 3 comments

$ThriceCommented(pid) := \Pi_{C1.pid} \sigma_{\substack{C1.pid=C2.pid=C3.pid \wedge \\ (C1.commenter \neq C2.commenter \vee C1.when \neq C2.when) \wedge \\ (C3.commenter \neq C2.commenter \vee C3.when \neq C2.when) \wedge \\ (C1.commenter \neq C3.commenter \vee C1.when \neq C3.when)}}$

$[\rho_{C1} Comment \times \rho_{C2} Comment \times \rho_{C3} Comment]$

– Posts where there was at least a change in sentiment

$AtLeastOneShift(pid) := \Pi_{pid} \sigma_{S1.pid=S2.pid \wedge S1.when<S2.when \wedge S1.sentiment!=S2.sentiment}$
$[\rho_{S1} Sent \times \rho_{S2} Sent]$

– Posts with exactlyOneShift

$OneShift(pid) := AtLeastOneShift - DoubleShiftingPosts$

– Remove any posts that had only 2 comments

$ShiftingPosts(pid) := ThriceCommented \cap OneShift$

–Add the fields we need for final answer. Done using a different approach where we rename columns so that we can use natural join and not have naming conflicts.

$Result(pid, uid, commenter, when, sentiment) := \Pi_{pid, uid, commenter, when, sentiment(text)}$
$[(\rho_{pid, uid, junk1, junk2, junk3} Post) \bowtie ShiftingPosts \bowtie Comment]$

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.)

   SOLUTION:

   – the set of posts where time of one of its comments is before time of the post is empty
   – Use the same technique as in Query 10 to rename and use natural join.

   $\sigma_{pTime > cTime}(\rho_{pid,junk1,pTime,junk2,junk3}Post) \bowtie (\rho_{pid,junk4,cTime,junk5}Comment) = \emptyset$

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

   SOLUTION:

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

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

   SOLUTION:

   $(\Pi pidPost - \Pi pidPIncludes) \cup (\rho_{pid}[\Pi_{sid}Story - \Pi sidSincludes]) = \emptyset$