University of Toronto
CSC343, Winter 2019

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 two post captions on every day of 2018. You may assume that there is at least one post on each day of a year.

   SOLUTION:

   – find posts in 2018, keep date and pid

   $2018posts(day, pid) := \Pi_{when.date,pid}\sigma_{when.year=\text{“2018”}}Post$

   – find all dates in 2018 (we can do this because of the assumption that there is at least one post on every day of the year)

   $2018dates(date) := \Pi_{when.date}\sigma_{when.year=\text{“2018”}}Post$

   – find hashtags with their posts from 2018

   $HT2018(date, tag, pid) := 2018posts \bowtie Hashtag$

– find those mentioned on 2 times on same date
$TwoMentions(date, tag) :=$

$$\Pi_{HT1.date, HT1.tag}$$
$$\sigma_{(HT1.tag=HT2.tag)\wedge(HT1.date=HT2.date)\wedge(HT1.pid>HT2.pid)}$$
$$[(\rho_{HT1}HT2018) \times (\rho_{HT2}HT2018)]$$

– make table of all dates in 2018 and all tags
$AllDaysTags(date, tag) := 2018dates \times \Pi_{tag}HT2018$

– find date,tag combos that did not have 2 mentions
$NotMentionedTwice(date, tag) := AllDatesTags - TwoMentions$

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

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
   $FollowsUser1(user1, user2, follower) := \Pi_{RF.user1, RF.user2, F.follower}\sigma_{RF.user1=F.followed}$
   $$[(\rho_{RF}ReciprocalFollowers) \times (\rho_{F}Follows)]$$

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

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

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

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

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

4. Find the user who has liked the least 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 user2 doesn't like ALL of user1 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 user1 doesn't like ALL of user2 posts

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

   – Backscratchers are all Reciprocal followers once we 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

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

   – 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. Report the name and email of the user who has gained the minimum number of new followers in 2018. If there is a tie, report them all.

   SOLUTION: Cannot be expressed by RA.

8. For each user who has ever put any comments, report their id and the id of the first and of the last post they commented on.

   SOLUTION:

   – Post comments that are not last

   $NotLastComment(commenter, pid) := \Pi_{P1.commenter,P1.pid}\sigma_{P1.commenter=P2.commenter \wedge P1.when<P2.when}$
   $[\rho_{P1}Comment \times \rho_{P2}Comment]$

   – Last post this user commented on

   $LastComment(commenter, last) := (\Pi_{commenter,pid}Comment) - NotLastComment$

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

   $NotFirstComment(commenter, pid, when) := \Pi_{P1.commenter,P2.pid,p2.when}\sigma_{P1.commenter=P2.commenter \wedge P1.when<P2}$
   $[\rho_{P1}Comment \times \rho_{P2}Comment]$

   $FirstComment(commenter, first) := \Pi_{commenter,pid}[Comment - NotFirstComment]$

   – Now natural join together to get all 3 columns

   $Result(commenter, first, last) := LastComment \bowtie FirstComment$

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 view on a story must occur after the date-time of the story itself. (Remember that you can compare two date-time attributes with simple $<$, $>=$ etc.)

   SOLUTION:

   – the set of stories where time of one of its views is before time of the story is empty

   $\sigma_{sTime>=vTime}(\rho_{sid,junk1,sTime,junk2}Story) \bowtie (\rho_{junk3,sid,vTime}Saw) = \emptyset$

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

   SOLUTION:

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