University of Toronto
CSC343, Fall 2017

Assignment 1: Zhihong Wang: 1002095207. Yecheng Song: 1002175754.

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".

   – uID of the all likers and posters.

   $LikersAndPosters(likers, posters) := \Pi_{liker,uid}(Likes \bowtie Post)$

   – Rename columns in LikersAndPosters.

   $LikersAndPosters(follower, followed) := LikersAndPosters(likers, posters)$

   – uID of all likers who has followed at least one of the posters they like.

   $LikersWhoFollowed(uID) := \Pi_{follower}(LikersAndPosters \bowtie Follows)$

   – uID of likers who never followed the posters they like.

   $NeverFollowLikers(uid) := \Pi_{uid}(User) - LikersWhoFollowed$

   – uID of the all viewers and the uID of the posters of the viewed posts.

   $ViewersAndPosters(viewers, posters) := \Pi_{viewerid,uid}(Saw \bowtie Story)$

   – Rename columns in ViewersAndPosters.

   $ViewersAndPosters(follower, followed) := ViewersAndPosters(viewers, posters)$

   – uID of all viewers who has followed at least one of the posters they viewed.

   $ViewersWhoFollowed(uID) := \Pi_{follower}(ViewersAndPosters \bowtie Follows)$

   – uID of viewers who never followed the posters they like.

   $NeverFollowViewers(uid) := \Pi_{uid}(User) - ViewersWhoFollowed$

   – uID of all users who never followed the posters they like.

   $NeverFollowUsers(uid) := NeverFollowViewers \cup NeverFollowLikers$

– (FINAL) username and description of all users who never followed the posters they like.

$NeverFollowUsersInfo(username, description) := \Pi_{name,about}(User \bowtie NeverFollowUsers)$

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 that are posted on every day of 2017 year.

$HashtagsIn2017(pid, tag, day) := \Pi_{pid,tag,when}\sigma_{when.year=2017}(Post \bowtie Hashtag)$

– Day list of 2017.

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

– Create "HashtagsIn2017_2"*by*"$HashtagsIn2017$".

$\text{HashtagsIn}2017_2(pid2, tag2, day2) := \rho_{HashtagsIn2017_2(pid2,tag2,day2)}HashtagsIn2017$

– Create "HashtagsIn2017_3"*by*"$HashtagsIn2017$".

$\text{HashtagsIn}2017_3(pid3, tag3, day3) := \rho_{HashtagsIn2017_3(pid3,tag3,day3)}HashtagsIn2017$

– Every hashtags that have been mentioned in at least three post captions on some day of 2017.

$ThreeTimesHashtags(day, tag) :=$

$\Pi_{day,tag}\sigma_{(day=day2=day3)\wedge(tag=tag2=tag3)\wedge(pid<pid2<pid3)}(HashtagsIn2017 \times HashtagsIn2017_2 \times HashtagsIn2017_3)$

– Tag list of 2017.

$2017TagList(tag) := \Pi_{tag}ThreeTimesHashtags$

– Test list of 2017.

$2017TestList(day, tag) := \Pi_{day,tag}(2017TimeList \times 2017TagList)$

– Not "Every hashtags that have been mentioned in at least three post captions on every day of 2017".

$Temp(day, tag) := \Pi_{day,tag}(2017TestList - ThreeTimesHashtags)$

– (FINAL) Every hashtags that have been mentioned in at least three post captions on every day of 2017.

$Result(tag) := \Pi_{tag}(ThreeTimesHashtags - Temp)$

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 "Follows2" by "Follows" as a copy of "Follows".

$Follows2(follower2, followed2, when) := \rho_{Follows2(follower2,followed2,when)} Follows$

– All distinct pairs of reciprocal followers.

$ReciprocalFollowers(recpId1, recpId2) :=$
$\Pi_{follower,followed} \sigma_{(follower=followed2) \wedge (followed=follower2) \wedge (follower<followed)} (Follows \times Follows2)$

– All followers who followed users with recpId1 in ReciprocalFollowers relation.

$AllFollowersOne(uid, recpId1, recpId2) :=$
$\Pi_{follower,recpId1,recpId2} \sigma_{(followed=recpId1)} (Follows \times ReciprocalFollower)$

– All followers who followed users with recpId2 in ReciprocalFollowers relation.

$AllFollowersTwo(uid, recpId1, recpId2) :=$
$\Pi_{follower,recpId1,recpId2} \sigma_{(followed=recpId2)} (Follows \times ReciprocalFollower)$

– Followers who followed both users (both of user with recpId1 and user with recpId2 in a pair) in reciprocal relations.

$CommonFollowers(uid, recpId1, recpId2) :=$
$\Pi_{uid,recpId1,recpId2} (AllFollowersOne \cap AllFollowersTwo)$

– Followers who followed one or both users in some reciprocal relation. (Followers who at least followed someone in that relation)

$AllFollowers(uid, recpId1, recpId2) := \Pi_{uid,recpId1,recpId2} (AllFollowersOne \cup AllFollowersTwo)$

– Followers who followed only one user in some reciprocal relation.

$OnlyFollowOne(uid, recpId1, recpId2) := AllFollowers - CommonFollowers$

– (FINAL) id, name and email of the users who only followed one of them but not the other, and the identifiers of the reciprocal followers.

$OnlyFollowOneInfo(recpId1, recpId2, uid, name, email) :=$
$\Pi_{recpId1,recpId2,uid,name,email} (OnlyFollowOne \bowtie Users)$

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.

   – Create "Follows2" as a copy of "Follows".
   $Follows2(follower2, followed2, when) := \rho_{Follows2(follower2,followed2,when)}Follows$

   – All distinct pairs of reciprocal followers.
   $ReciprocalFollowersPairs(recpId1, recpId2) :=$
   $\Pi_{follower,followed}\sigma_{(follower=followed2)\wedge(followed=follower2)\wedge(follower<followed)}(Follows \times Follows2)$

   – Posts liked by "recpId1".
   $recpId1Liked(recpId1, recpId2, post) :=$
   $\Pi_{recpId1,recpId2,pid}\sigma_{recpId1=liker}(ReciprocalFollowersPairs \times Likes)$

   – Posts liked by "recpId2".
   $recpId2Liked(recpId1, recpId2, post) :=$
   $\Pi_{recpId1,recpId2,pid}\sigma_{recpId2=liker}(ReciprocalFollowersPairs \times Likes)$

   – All posts by "recpId1".
   $recpId1Posted(recpId1, recpId2, post) :=$
   $\Pi_{recpId1,recpId2,pid}\sigma_{uid=recpId1}(Post \times ReciprocalFollowersPairs)$

   – All posts by "recpId2".
   $recpId2Posted(recpId1, recpId2, post) :=$
   $\Pi_{recpId1,recpId2,pid}\sigma_{uid=recpId2}(Post \times ReciprocalFollowersPairs)$

   – Users in "recpId2" who do not like all of the posts of "recpId1".
   $recpId2NotAll(recpId1, recpId2) := \Pi_{recpId1,recpId2}(recpId1Posted - recpId2Liked)$

   – Users in "recpId1" who do not like all of the posts of "recpId2".
   $recpId2NotAll(recpId1, recpId2) := \Pi_{recpId1,recpId2}(recpId2Posted - recpId1Liked)$

   – "recpId1" and "recpId2" who like all posts of each others.
   $Backscratchers(recpId1, recpId2) :=$
   $\Pi_{recpId1,recpId2}(ReciprocalFollowersPairs - recpId1NotAll - recpId2NotAll)$

– All followers of "recpId1" in backscratchers relationship.
$Follower1(follower) := \Pi_{follower}\sigma_{followed=recpId1}(Follows \times Backscratchers)$

– All followers of "recpId2" in backscratchers relationship.
$Follower2(follower) := \Pi_{follower}\sigma_{followed=recpId2}(Follows \times Backscratchers)$

– (FINAL) All followers of both Backscratchers.
$BackscratchersFollowers(follower) := (Follower1 \cap Follower2)$

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.

   – pid/sid of all stories and posts, along with the uids of the posters and the time they are posted.(pid and sid are renamed to pid now)

   $PostsAndStories(pid, poster, time) := \Pi_{pid,uid,when}(Post) \cup \Pi_{sid,uid,when}(Story)$

   – Create a copy of PostsAndStories(pid, poster, time) named PostsAndStories2(pid2, poster2, time2) for cross product

   $PostsAndStories2(pid2, poster2, time2) := \rho_{PostsAndStories2(pid2,poster2,time2)} PostsAndStories$

   – pid/sid of the stories and posts that are not the most recent for every user, along with the uids of thier posters and the time they are posted.

   $PostsToIgnore(pid, poster, time) :=$
   $\Pi_{pid,poster,time}\sigma_{(poster=poster2)\wedge(time<time2)}(PostsAndStories \times PostsAndStories2)$

   – pid/sid of the most recent stories and posts of every poster, along with the uids of their posters and the time they are posted. The uids of the posters are named followed, for the purpose of natural joint later.

   $MostRecentPosts(pid, followed, time) := PostsAndStories - PostsToIgnore$

   – All follower and followed relationship, including the time of the most recent act of the followed poster.

   $FollowsAndRecentAct(follower, followed, time) :=$
   $\Pi_{follower,followed,time}(MostRecenPosts \bowtie Follows)$

   – Create a copy of FollowsAndRecentAct(follower, followed, time) named FollowsAndRecentAct2(follower2, followed2, time2) for cross product

   $PostsAndStories2(pid2, poster2, time2) := \rho_{PostsAndStories2(pid2,poster2,time2)} Follows$

– All followed realationship, in which every followed person is NOT the most recently active user to the follower, including the time of the most recent act of the followed poster.

$MostRecentFollows(follower, followed, time) :=$
$\Pi_{follower,followed,time}\sigma_{(follower=follower2)\wedge(time<time2)}$
$(FollowsAndRecentAct \times FollowsAndRecentAct2)$

– All followed realationship, in which every followed person IS the most recently active user to the follower, including the time of the most recent act of the followed poster.

$MostRecentFollows(follower, followed, time) :=$
$FollowsAndRecentAct - MostRecentFollows$

– The name of every user, and the uid of the most recently active user they follow, and the date of the most-recent activity.

$UsernameAndMostRecentUid(username, posteruid, time) :=$
$\Pi_{name,followed,time}\sigma_{follower=uid}(User \times MostRecentFollows)$

– (FINAL) The name of every user, and the name and email address of the most recently active user they follow, and the date of the most-recent activity.

$UserAndMostRecentPoster(username, poster_name, email, time) :=$
$\Pi_{username,name,email,time}\sigma_{posteruid=uid}(UsernameAndMostRecentUid \times MostRecentFollows)$

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.

– Record of all dates.
$Record(pid, whenpost, whenlike, liker) := \Pi_{pid,Post.when,Likes.when,liker}(Post \times Likes)$

– The copy of "Record".
$Record2(pid2, whenpost2, whenlike2, liker2) := \rho_{Record2(pid2,whenpost2,whenlike2,liker)}Record$

– All not-in-order dates.
$UnorderRecord(pid, whenpost, whenlike, liker) :=$
$\Pi_{pid,whenpost,whenlike,liker}\sigma_{(liker=liker2)\wedge(whenpost>whenpost2)\wedge(whenlike<whenlike2)}(Record \times Record2)$

– All likers in "Record".
$AllLikers(liker) := \Pi_{liker}Record$

– All likers in "UnorderRecord".
$UnorderLikers(liker) := \Pi_{liker}UnorderRecord$

– Likers in order.
$OrderLikers(liker) := \Pi_{liker}(AllLikers - UnorderLikers)$

– (FINAL) Likers in order and emails.
$Result(liker, email) := \Pi_{liker,email}\sigma_{liker=uid}(OrderLikers \times User)$

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.

– Make a copy of the Saw(viewerid, sid, when) relationship named Saw2(viewerid, sid2, when2) for the natural joint later.

$Saw2(viewerid, sid2, when2) := \rho_{Saw2(viewerid,sid2,when2)}Saw$

– the viewerid, sid number and time of the posts/stories that are not most recently read by that viewer.

$NotLastlyViewed(viewerid, sid, when) := \Pi_{viewerid,sid,when}\sigma_{when<when2}(Saw \bowtie Saw2)$

– the viewerid, sid number and time of the posts/stories that are most recently read by that viewer.

$LastlyViewed(viewerid, sid, when) := Saw - NotLastlyViewed$

– the viewerid, sid number and time of the posts/stories that are not firstly read by that viewer.

$NotFirstlyViewed(viewerid, sid, when) := \Pi_{viewerid,sid,when}\sigma_{when>when2}(Saw \bowtie Saw2)$

– the viewerid, sid number and time of the posts/stories that are firstly read by that viewer.

$FirstlyViewed(viewerid, sid, when) := Saw - NotFirstlyViewed$

– (FINAL) the viewerid and sid number of the posts/stories that are firstly OR most recently read by that viewer.

$FirstlyOrLastly(viewerid, sid) := \Pi_{viewerid,sid}(LastlyViewed \cup FirstlyViewed)$

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.

    – The first copy of "Comment".

    $Comment1(pid1, commenter1, when1, text1) := \rho_{Comment1(pid1,commenter1,when1,text1)}Comment$

    – The second copy of "Comment".

    $Comment2(pid2, commenter2, when2, text2) := \rho_{Comment2(pid2,commenter2,when2,text2)}Comment$

    – Those posts have more then 3 comments.

    $ThreeCommentsPosts(pid) :=$
    $\Pi_{pid}\sigma_{(pid=pid1=pid2)\wedge(commenter<commenter1<commenter2)}(Comment \times Comment1 \times Comment2)$

    – All comments in posts which have more then 3 comments.

    $Comments3(pid, commenter, when, text) :=$
    $\Pi_{pid,commenter,when,text}Comment \bowtie ThreeCommentsPosts$

    – All positive comments in posts which have more then 3 comments.

    $PosComments(pid1, commenter1, when1) :=$
    $\Pi_{pid,commenter,when}\sigma_{sentiment(text)="positive"}Comments3$

    – All negative comments in posts which have more then 3 comments.

    $NegComments(pid2, commenter2, when2) :=$
    $\Pi_{pid,commenter,when}\sigma_{sentiment(text)="negative"}Comments3$

    – All "more than 3 comments" posts which have occurred "positive shift to negative comments".

    $PosToNegPosts(pid) := \Pi_{pid}\sigma_{(pid1=pid2)\wedge(when1<when2)}(PosComments \times NegComments)$

    – All "more than 3 comments" posts which have occurred "negative shift to positive comments".

    $NegToPosPosts(pid) := \Pi_{pid}\sigma_{(pid1=pid2)\wedge(when2<when1)}(PosComments \times NegComments)$

– All posts that not satisfy the definition of "Sentiment shift" in this question because there are more than one shift happened.

$MoreThanOneShiftPosts(pid) := NegToPosPosts \cap PosToNegPosts$

– All posts that satisfy the definition of "Sentiment shift" in this question because there are only one shift happened.

$OneShiftPosts(pid) := ThreeCommentsPosts - MoreThanOneShiftPosts$

– All comments that satisfy the definition of "Sentiment shift" in this question.

$OneShiftComments(pid, commenter, when, text) := OneShiftPosts \bowtie Comments3$

– (FINAL) 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 commenters id,the date of their comment and its sentiment.

$Result(pid, owner, commenter, when, sentiment) :=$
$\Pi_{pid,uid,commenter,when,sentiment(text)}(Post \bowtie OneShiftComments)$

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.)
   $\Pi_{pid}\sigma_{Post.when>=Comment.when}Post \bowtie Comment = \emptyset$

2. Each user can have at most one current story.
   $Story2(sid2, uid2, when2, current2) := \rho_{Story2(sid2,uid2,when2,current2)}Story$
   $\Pi_{uid}\sigma_{(uid=uid2)\wedge(current="yes")\wedge(current2="yes")\wedge(sid<sid2)}(Story \times Story2) = \emptyset$

3. Every post must include at least one picture or one video and so must every story.
   $NoContentsPost(id) := \Pi_{pid}Post - \Pi_{pid}PIncludes$
   $NoContentsStory(id) := \Pi_{sid}Story - \Pi_{sid}SIncludes$
   $NoContentsPost + NoContentsStory = \emptyset$