**Tianshu Zhu**

**1002111225**

**Part1**

**Q1**

“Liker” liked some post of “liked”

Liked(liker, liked):=

ΠLikes.liker, Post.pid(σ*Likes.pid = Post.pid* (Likes × Post))

“Viewer” viewed some story of “viewed”

Viewed(viewer, viewed):=

ΠSaw.viewerid, Story.sid(σ*Saw.sid = Story.sid* (Saw × Story))

“Uid1” liked some post or viewed some story of “uid2” and

“Uid1” did not followed “uid2”

NotSat(uid1, uid2):=

Liked ∪ Viewed − Πfollower, followed(Follows)

“Uid” never liked some post or viewed some story of user whom he did not follow

Sat(uid):=

Πuid(User) − Πuid1(NotSat)

Report

Result(username, description):=

ΠUser.name, User.about(σ*User.uid = Sat.uid* (User × Sat))

**Q2**

“Pid” is posted in 2017 along with its posted date and tag

PostTag17(pid, date, tag):=

ΠPost.pid, when, Hashtag.tag(σ*when.year = 2017 ∧ Post.pid = Hashtag.pid*(Post × Hashtag))

Join of all dates and all tags of 2017

All(date, tag):=

Πwhen.date(σ*when.year = 2017* (Post)) × Πtag(PostTag17)

“Tag” was mentioned in 2017 but was not mentioned everyday in 2017

NotEveryday(tag):=

Πtag(All − Πdate, tag(PostTag17))

“Tag” was mentioned everyday in 2017

Everyday(tag):=

Πtag(PostTag17) − NotEveryday

“Tag” was mentioned at least 3 times everyday in 2017

result(tag):=

ΠEveryday.tag(σ*(p1.tag = p2.tag= p3.tag = Everyday.tag)*  *∧ (p1.tag ≠ p2.tag ∧ p1.tag ≠ p3.tag ∧ p2.tag ≠ p3.tag)*(ρp1 (PostTag17) × ρp2 (PostTag17) × ρp3 (PostTag17) × Everyday))

**Q3**

“Uid1”, “uid2” are reciprocal follower and “uid1” > “uid2”

ReciprocalFollower(uid1, uid2):=

Πf1.followed, f1.follower(σ*(f1.follower = f2.followed) ∧ (f1.followed = f2.follower) ∧ (f1.follower > f1.followed)*(ρf2 (Follows) × ρf1 (Follows)))

“Follower” followed “uid1”

uid1Follower(uid1, uid2, follower):=

Πuid1, uid2, follower(σ*uid1 = followed* (ReciprocalFollower × Follows))

“Follower” followed “uid2”

uid2Follower(uid1, uid2, follower):=

Πuid1, uid2, follower(σ*uid2 = followed* (ReciprocalFollower × Follows))

“Uid1”, “uid2” are reciprocal follower, “follower” are uncommon follower of “uid1”, “uid2”

UncommonFollower(uid1, uid2, follower):=

(uid1Follower ∪ uid2Follower) − (uid1Follower ∩ uid2Follower)

Report

result(uid1, uid2, follower, name, email):=

Πuid1, uid2, follower, name, email(σ*follower = uid* (UncommonFollower × User))

**Q4**

Can not be expressed

**Q5**

“Uid1”, “uid2” are reciprocal follower and “uid1” > “uid2”

ReciprocalFollower(uid1, uid2):=

Πf1.followed, f1.follower(σ*(f1.follower = f2.followed) ∧ (f1.followed = f2.follower) ∧ (f1.follower > f1.followed)*(ρf2 (Follows) × ρf1 (Follows)))

“Liker” liked some post “pid” of “liked”

Liked(liker, liked, pid):=

Πliker, Post.uid, Post.pid(σ*Likes.pid = Post.pid* (Likes × Post))

Join all liker in likes with all (uid, pid) in Post

All(liker, liked, pid):=

Πliker(Likes) × Πpid, uid(Post)

“Liker” did not like every post of “liked”

NotLikedEvery(liker, liked):=

All - Liked

“Liker” liked every post of “liked”

LikedEvery(liker, liked):=

Πliker, liked(Liked) - Πliker, liked(NotLikedEvery)

“Uid1”, “uid2” liked every post of eachother

ReciprocalLiker(uid1, uid2):=

Πr1.liker, r1.liked(σ*r1.liker = r2.liked ∧ r1.liked = r2.liker*  (ρr1 (LikedEvery) × ρr2 (LikedEvery)))

“Uid1”, “uid2” are backscratchers

Backscratchers(uid1, uid2):=

ReciprocalFollower ∩ ReciprocalLiker

**Q6**

“When” is the datetime of some activity of “uid”

WhenActivity(uid, when):=

Πuid.when(Post) ∪ Πuid, when(Story)

“When” is datetime of some activity of “followed”, who is followed by “follower” with “name”

WhenFollowedActivity(name, follower, followed, when):=

ΠUser.name, User.uid, *WhenActivity.uid, WhenActivity.when* (σ*User.uid = Follows.follower ∧ Follows.followed = WhenActivity.uid*  (User × Follows ×WhenActivity ))

For each “follower” in WhenFollowedActivity, tuple of that follower with most recent “when” is removed

NotMostRecent(name, follower, followed, when):=

Πr1.name, r1.follower, r1.followed, r1.when(σ*r1.follower = r2.follower ∧ r1.when < r2.when* (ρr1 (WhenFollowedActivity) × ρr2 (WhenFollowedActivity)))

For each “follower” WhenFollowedActivity, get the tuple of that follower with most recent “when”

MostRecent(name, follower, followed, when):=

WhenFollowedActivity - NotMostRecent

Report user with the most recent user he followed

Result(followerName, followedName, followedEmail, date):=

Π*MostRecent.name, User.name, User.email, MostRecent.when.date* (σ*MostRecent.followed = User.uid* (MostRecent × User))

**Q7**

“LikeDate” is date when “uid” liked some post which was posted on “postDate”

DateLikePost(uid, likeDate, postDate):=

ΠLikes.liker, Likes.when.date, Post.when.date(σ*Likes.pid = Post.pid* (Likes × Post))

“Uid” did not satisfy question requirement:

“users who have always liked posts in the same order as the order in which they were posted”

NotSat(uid):=

Πr1.uid(σ*(r1.likeDate > r2.likeDate) ∧ (r1.postDate < r2.postDate) ∧ (r1.uid = r2.uid)* (ρr1 (DateLikePost) × ρr2 (DateLikePost)))

“Uid” satisfied question requirement

Sat(uid):=

Πuid (User) - NotSat

Report

Result(name, email):=

Πname, email(σ*User.uid = Sat.uid* (Sat × User))

**Q8**

Can not be expressed

**Q9**

“Sid” was not the last story that “uid” viewed

NotLastStory(viewerid, sid):=

Πs1.viewerid, s1.when(σ*(s1.viewerid = s2.viewerid) ∧ (s1.when < s2.when)* (ρs1 (Saw) × ρs2 (Saw)))

“Sid” was the last story that “uid” viewed

LastStory(viewerid, sid):=

Πviewerid, sid(Saw) − NotLastStory

“Sid” was not the first story that “uid” viewed

NotFirstStory(viewerid, sid):=

Πs1.viewerid, s1.when(σ*(s1.viewerid = s2.viewerid) ∧ (s1.when > s2.when)* (ρs1 (Saw) × ρs2 (Saw)))

“Sid” was the first story that “uid” viewed

FirstStory(viewerid, sid):=

Πviewerid, sid(Saw) − NotFirstStory

report uid and the id of the first and of the last story he have seen

result(viewerid, firstSid, lastSid):=

Π*FirstStory.viewerid, FirstStory.sid, LastStory.sid* (σ*FirstStory.viewerid = LastStory.viewerid* (FirstStory × LastStory))

**Q10**

“Pid” has at least 3 different comments

ThreeComments(pid, commentor, when, sentiment):=

Πc1.pid, c1.commentor, c1.when, sentiment(c1.text)(σ*(c1.pid = c2.pid = c3.pid) ∧ (c1.commentor ≠ c2.commentor ∨c1.when ≠ c2.when) ∧ (c1.commentor ≠ c3.commentor ∨c1.when ≠ c3.when) ∧ (c2.commentor ≠ c3.commentor ∨c2.when ≠ c3.when)* (ρc1 (Comment) × ρc2 (Comment) × ρc3 (Comment)))

“Pid” in ThreeComments, and pid has comments of at least 2 different “when”

LeastTwoWhen(pid, commentor, when, sentiment):=

Πr1.pid, p1.commentor, r1.when, r1.sentiment(σ*(r1.pid = r2.pid) ∧ (r1.when ≠ r2.when)* (ρr1 (ThreeComments) × ρr2 (ThreeComments)))

“Pid” in ThreeComments, and pid has comments of at least 3 different “when”

LeastThreeWhen(pid, commentor, when, sentiment):=

Πr1.pid, p1.commentor, r1.when, r1.sentiment(σ*(r1.pid = r2.pid = r3.pid) ∧ (r1.when < r2.when < r3.when)* (ρr1 (ThreeComments) × ρr2 (ThreeComments) × ρr3 (ThreeComments)))

“Pid” in ThreeComments, and pid has comments all of the same “when”.

These “pid”s can not have shift

OneWhen(pid, commentor, when, sentiment):=

ThreeComments - LeastTwoWhen

OneWhenNoShift:= OneWhen

“Pid” in ThreeComments, and pid has comments of exactly 2 different “when”

TwoWhen(pid, commentor, when, sentiment):=

LeastThreeWhen - LeastTwoWhen

When there are at least 3 different when, get “pid” with no shift

LeastThreeWhenNoShift(pid, commentor, when, sentiment):=

Πr1.pid, p1.commentor, r1.when, r1.sentiment(σ*(r1.pid = r2.pid = r3.pid) ∧ (r1.when < r2.when < r3.when) ∧ (r1.sentiment ≠ r2.sentiment ∧ r2.sentiment ≠ r3.sentiment)* (ρr1 (LeastThreeWhen) × ρr2 (LeastThreeWhen) × ρr3 (LeastThreeWhen)))

When there are exactly 2 different when, get “pid” with no shift

TwoWhenNoShift(pid, commentor, when, sentiment):=

Πr1.pid, p1.commentor, r1.when, r1.sentiment(σ*(r1.pid = r2.pid) ∧ (r1.when ≠ r2.when) ∧ (r1.sentiment = r2.sentiment)* (ρr1 (TwoWhen) × ρr2 (TwoWhen)))

“Pid” had at least three comments and for which there has been a sentiment shift over time

Shift(pid, commentor, when, sentiment):=

(((ThreeComments - OneWhenNoShift) - TwoWhenNoShift) - LeastThreeWhenNoShift)

Report

Result(uid, pid, commentor, date, sentiment):=

ΠPost.pid, Post.uid, Shift.commentor, Shift.when.date, Shift.sentiment(σ*Post.pid = Shift.pid* (Post × Shift))

**Part2**

**Q1**

σ*(Comment.pid = Post.pid) ∧ (comment.when ≤ Post.when)* (Comment × Post) = ∅

**Q2**

σ*(s1.uid = s2.uid) ∧ (s1.sid ≠ s2.sid) ∧ (s1.current = true ∧ s2.current = true)* (ρs1 (Story) × ρs2 (Story)) = ∅

**Q3**

Πpid(Post) - Πpid(PIncludes) = ∅

Πsid(Story) - Πsid(SIncludes) = ∅