Ruijie Sun 1003326046
Wenjie Hao 1002183059

Relational Algebra Symbols

Unary operators on relations:

$\Pi_{x, y, z}(R)$

$\sigma_{condition}(R)$

$\rho_{\mathrm{New}}(R)$

$\rho_{\mathrm{New}(a, b, c)}(R)$

Binary operators on relations:

$R \times S$

$R \bowtie S$

$R \bowtie_{condition} S$

$R \cup S$

$R \cap S$

$R - S$

Logical operators: $\leftarrow \wedge \vee$

Assignment:

$\mathrm{New}(a, b, c) := R$

Ruijie Sun 1003326046
Wenjie Hao 1002183059

Part1:

1. Find all the users who have never sent a message, but who have been sent at least one message. The message may have been sent to the user or to a group that the user belongs to. Report each user id.

$UserSentAtLeast(uid) = \Pi_{from} (Message)$

$UserNeverSent(uid) = (\Pi_{uid} (User)) - UserSentAtLeast$

$UserDirectReceived(uid) = \Pi_{User.uid} (\sigma_{Message.to = User.uid} (Message \times User))$

$DirectGroupReceived(gid) = \Pi_{Group.gid} (\sigma_{Message.to = Group.gid} (Message \times Group))$

$UserIntermediateReceived(uid) = \Pi_{User.uid} (DirectGroupReceived \bowtie User)$

$Answer(uid) = (UserIntermediateReceived \cup UserDirectReceived) \cap UserNeverSent$

2. Net neutrality is dead, so EVL ISP wants to slow the service of poor users (users who do not use the app enough). To do this, find the users (and return their uid) who sent two or fewer messages in 2017.

$AtLeastThree(uid) = \Pi_{from} (\sigma_{M1.from = M2.from = M3.from \wedge M1.mid > M2.mid > M3.mid \wedge M1.time.year = M2.time.year = M3.time.year = 2017} (\rho_{M1} (Message) \times \rho_{M2} (Message) \times \rho_{M3}(Message)))$

$Answer(uid) = \Pi_{uid} (User) - AtLeastThree$

3. Find the largest group. Report the group id. If there is a tie, report them all.

Can not be expressed.

4. Find privacy fanatics, that is, any user who has all her privacy settings set to none and who has never sent a message to another user who has privacy settings different than her own (meaning different than all none). Note that a private user (settings are all none) who has never sent a message would be considered a privacy fanatic. Return the user's uid and name.

PrivateUsers(uid) =Π_{uid} ($\sigma_{lastseen\ =\ None\ \wedge photo\ =\ None\ \wedge profile\ =\ None}$ (Privacy))

NonPrivateUsers(uid) = (Π_{uid} (User)) – PrivateUsers

Intermediate1(uid) = Π_{from} ($\sigma_{Message.to\ =\ NonPrivateUsers.uid}$ (Message × NonPrivateUsers))

TargetInfoThroughtGroup(from,uid) = $\Pi_{Message\cdot from,Group.uid}$ ($\sigma_{Message.to\ =\ Group.gid}$ (Message × Group))

Intermediate2(uid) = $\Pi_{TargetInfoThroughtGroup.from}$ ($\sigma_{TarfetInfoThroughtGroup.uid=NonPrivateUsers.uid}$ (TargetInfoThroughtGroup × NonPrivateUsers))

Answer(uid) = PrivateUsers – (Intermediate1 ∪ Intermediate2)

5. Consider only users whose privacy settings state that everyone may see their lastSeen time (lastSeen = everyone). Among such users, report the uid, name and lastSeen of the user(s) whose lastSeen time is the most recent. Since times are ordered, the most recent time is the largest time value. If there are ties, report all users. These users have the most recent public lastSeen time.

TargetUsers(uid) = Π_{uid} ($\sigma_{lastseen\ =\ everyone}$ (Privacy))

Middle(uid, name, lastseen) = $\Pi_{uid,name,lastseen}$ ($\sigma_{User.uid\ =\ TargetUsers.uid}$

(User × TargetUsers))

NotMax = Π_{uid} ($\sigma_{M1.lastseen>M2.lastseen}$ (ρ_{M1} (Middle) ×ρ_{M2} (Middle))

))

Max(uid) = TargetUsers – Notmax

Answer(uid, name, lastseen) = $\Pi_{uid,name,lastseen}$ (Max ⋈ User)

6. A user's contact list can be sorted by the start time. Find users who send their first direct message to a contact in the same order as the contact list order. So if Sue is Pat's oldest contact and Jo is the second oldest contact, then Pat's first direct message to Sue happens before her first direct message to Jo and so on for all contacts. Include users with empty contact lists. Return user's uid.

MessageWithTime(user, contact, time) = $\Pi_{from,to,time}$ (Message)

NotFirst(user, contact, time) = $\Pi_{M1.user,\ M1.contact,\ M1.time}$ ($\sigma_{M1.user=M2.user \wedge M1.contact=M2.contact \wedge M1.time > M2.time}$ (ρ_{M1} (MessageWithTime) $\times \rho_{M2}$ (MessageWithTime))

))

ContactFirstMessage(user, contact, first) = MessageWithTime – NotFirst

ContactWithFirstMessage(user,contact,first,start) = $\Pi_{Contact.user, Contact.contact, ContactFirstMessage.fisrt, Contact.start}$($\sigma_{ContactFirstMessage.user = Contact.user \wedge ContactFirstMessage.contact = Contact.contact}$ (ContactFirstMessage × Contact))

// To find the users who violate the rules. Assume that there is no the case where a user send messages to two of his contacts at same time or a user know two contacts at same time.

Not(uid) = $\Pi_{A1.user}$ ($\sigma_{A1.user = A2.user \wedge A1.contact != A2.contact \wedge A1.first > A2.first \wedge A1.start < A2.start}$ (ρ_{A1} (ContactWithFirstMessage) $\times \rho_{A2}$ (ContactWithFirstMessage))

Answer = Π_{uid} (User) - Not

7. Return all pairs of users with the same name. Return the two uids and the common name. Return each pair only once. (For example, if user 1 and user 2 are both named ’Pat’, then return either [1, 2, ’Pat’] or [2, 1, ’Pat’] but not both).

Answer(uid1, uid2, name)

= $\Pi_{U1.uid, U2.uid, U1.name}$ $\sigma_{U1.uid < U2.uid \wedge U1.name = U2.name}$ (ρ_{U1} (User) $\times$ ρ_{U2} (User))

8. For each user and contact, report the time that the first direct message was sent from the user to the contact and the time the last direct message was sent. Return the uid of the user (in an attribute named user) and the contact (in an attribute named contact) and the first time (earliest) (in an attribute named first) and last (most recent) time (in an attribute named last). If a user has not sent any direct messages to a contact then include the user and contact with the value 0 for both the first and last times.

AllPairs(user, contact) = $\Pi_{user, contact}$ Contact

MessageInfo(from, to, time) = $\Pi_{from, to, time}$ Message

TrueContacts(user, contact, time) = $\Pi_{AllPairs.user, AllPairs.contact, MessageInfo.time}$ $\sigma_{AllPairs.user =MessageInfo.from \wedge AllPairs.contact = MessageInfo.to}$ (AllPairs × MessageInfo)

NotMinContacts(user, contact, time)

= $\Pi_{T1.user, T1.contact, T1.time}$ $\sigma_{T1.time > T2.time \wedge T1.user = T2.user \wedge T1.contact=T2.contact}$ (ρ_{T1} (TrueContacts) × ρ_{T2} (TrueContacts))

MinContacts(user, contact, time) = TrueContacts – NotMinContacts

NotMaxContacts(user, contact, time)

= $\Pi_{\text{T3.user, T3.contact, T3.time}}\ \sigma_{\text{T3.time < T4.time ^ T3.user = T4.user ^ T3.contact=T4.contact}}$ (ρ_{T3} (TrueContacts) × ρ_{T4} (TrueContacts))

MaxContacts(user, contact, time) = TrueContacts – NotMaxContacts

AnswerPartA(user, contact, first, last) = (MinContacts) $\bowtie$ MaxContacts)

AnswerPartB(user, contact, first, last) =$\Pi_{\text{user, contact, 0, 0}}$ (AllPairs - ($\Pi_{\text{user, contact}}$ TrueContacts))

Answer(user, contact, first, last) = AnswerPartA ∪ AnswerPartB

9. A 'spammer' is a user who posts unwanted direct messages that are not read. A spammer must have sent at least direct message (so this message will appear in the Status relation). Because users may not be aware that someone is a spammer, they may read some of their initial messages. However, once they decide a certain user is a spammer, the receivers stop reading all messages from the spammer. This means that for a user who is sent a direct message from a spammer there are no delivered messages with a time that is earlier than any read message from the spammer. Return the spammer's user id and all their privacy settings (Privacy.lastSeen, Privacy.photo, Privacy.profile). Do not consider groups for this question. Only consider direct messages sent from a user to another single user (not to a group).

AllMessages(mid, from, to, time) = =$\Pi_{\text{mid, from, to, time}}$ (Message)

AllMessageswithStatus(mid, from, to, time, status) = $\Pi_{\text{Status.mid, AllMessages.from, Status.uid, AllMessages.time, Status.status}}$ ($\sigma_{\text{Status.mid=AllMessages.mid ^ Status.uid =AllMessages.to}}$ (AllMessages × Status))

NonSpammerPartA(uid) = $\Pi_{\text{from}}\ \sigma_{\text{A1.from =A2.from ^ A1.to = A2.to} \wedge A1.time < \text{A2.time} \wedge A1.status = \text{"}delivered\text{"} \wedge A2.status = \text{“read”}}$ (ρ_{A1} (AllMessagewithsStatus) × ρ_{A2} (AllMessageswithStatus))

AnswerPartA (uid) = (Π_{from} (AllMessageswithStatus)) - NonSpammerPartA

UserBeenRead(uid) = Π_{from} ($\sigma_{\text{status= “read”}}$ (AllMessageswithStatus))

UserNeverBeenRead(uid) = Π_{from}(Message) – UserBeenRead

Answer = AnswerPartA ∪ UserNeverBeenRead

Part 2

1.

Message[to] – Group[gid] – User[uid] = ∅

2.

DirectContactInfo(user, contact, time, start) = $\Pi_{Message.from,Message.to,Message.time,Contact.start}(\sigma_{Message.from=Contact.user \wedge Message.to=Contact.contact}((\Pi_{from, to, time}(Message)) \times Contact))$

Intermediate1 (user, contact, time) = $\Pi\text{user,contact,time}((\rho_{Middle(user,gid,time)}(\Pi_{from,to,time}(Message))) \bowtie (\rho_{G(gid,contact)}Group))$

IndirectContactInfo(user, contact, time, start) = $\Pi\text{user,contact,time,start}(\text{Intermediate1} \bowtie Contact)$

TotalContactInfo(user, contact, time, start) = DirectContactInfo ∪ IndirectContactInfo

$\rho_{Mess(user,contact)}(\Pi_{from, to}(Message) - \Pi\text{user, contact}(TotalContactInfo) = \emptyset$

$\sigma_{time < start}(TotalContactInfo) = \emptyset$

3. Can not be expressed.

4.

MessageInfo(mid, to) = $\Pi_{mid, to}(Message)$

IndirectUsers(mid, uid) = $\Pi_{\text{MessageInfo.mid, Group.uid}}\ \sigma_{\text{Group.uid=MessageInfo.to}}$ (Message × Group)

DirectUsers(mid, uid) = $\Pi_{\text{MessageInfo.mid, User.uid}}\ \sigma_{\text{User.uid=MessageInfo.to}}$ (Message × Group)

UserMessages(mid, uid) = IndirectUsers $\cup$ DirectUsers

StatusInfo(mid, uid) = $\Pi_{\text{mid, to}}$ Status

StatusInfo – UserMessages = $\emptyset$