University of Toronto

CSC343, Winter 2018

# Assignment 1:
# Alexandra Teachman (1001383166) Jay Zuo (1002530077)

## Part 1: Queries

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.

$$
\begin{aligned}
&All := Users\\
&NeverSentMessage := All - (\sigma_{Message.from=User.uid}(Message \times User)\\
&RecievedUser := \sigma_{Message.to=User.uid}(Message \times User)\\
&RecievedGroup := \sigma_{Message.to=Group.gid}(Message \times Group)\\
&UsersInRecievedGroup := \sigma_{User.uid=Group.uid}(RecievedGroup \times User)\\
&AllRecieved := (RecievedUser \bowtie UsersInRecievedGroup)\\
&Answer := \Pi_{NeverSentMessage.uid}(NeverSentMessage \cap AllRecieved)
\end{aligned}
$$

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.

$$All := \Pi from \sigma_{time.year=2017} Messsage$$

$$Atleast3 :=$$

$$\Pi M1.from \sigma_{\substack{M1.from=M2.from\wedge\\ M1.from=M3.from\wedge\\ M1.time.year=2017\wedge\\ M2.time.year=2017\wedge\\ M3.time.year=2017\wedge\\ M1.mid!=M2.mid\wedge\\ M2.mid!=M3.mid\wedge\\ M3.mid!=M1.mid}}[(\rho_{M1(mid,from,time)}Message)\times(\rho_{M2(mid,from,time)}Message\times$$

$$\rho_{M3(mid,from,time)}Message)]$$

$$Answer := All - Atleast3$$

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

   Cannot 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 users *uid* and `name`.

$PrivateUsers := \Pi uid \sigma_{lastSeen=none \wedge photo=none \wedge profile=none} Privacy$

$NonPrivateUsers := \Pi uid Privacy - PrivateUsers$

$MessagesFromPrivate := PrivatesUsers \bowtie_{uid=from} Message$

$MessagesToPrivate := PrivateUsers \sigma_{uid=to} MessagesFromPrivate$

$MessagesToNonprivate := NonPrivateUsers \sigma_{uid=to} messagesFromPrivate$

$Answer := \Pi uid, name MessagesToPrivate - \Pi uid, name MessagesToNonPrivate$

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.

   $PublicUsers := \Pi_{uid}(\sigma_{lastSeen=everyone}(Privacy))$

   $UserList := \sigma_{PublicUsers.uid=User.uid}(PublicUsers \times User)$

   $A := UserList$

   $B := UserList$

   $MostRecent := \Pi_{lastSeen}(Users) - (\Pi_{A.lastSeen}(A \bowtie_{A.lastSeen>B.lastSeen} B)$

   $Answer := \Pi_{uid,name,lastSeen}(\sigma_{MostRecent.lastSeen=Users.lastSeen}(MostRecent \times User))$

6. A users 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 Pats oldest contact and Jo is the second oldest contact, then Pats 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 users uid.

   $EmptyUsers := \rho_{U1(user)}(\Pi_{uid}(User)) - \Pi_{user}(Contact)$

   $C1 := Contact$

   $C2 := Contact$

   $M1 := Message$

   $M2 := Message$

   $$OrderedUsers := \sigma_{\substack{C1.user=C2.user \wedge C1.start<C2.start \\ \wedge M1.to=C1.contact \wedge M2.to=C2.contact \\ \wedge C1.user=M1.from \wedge C1.user=M2.from \\ \wedge M1.time<M2.time}} (C1 \times C2 \times M1 \times M2)$$

   $Answer := \Pi_{user}(EmptyUsers \times OrderedUsers)$

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).

   $U1 := users$

   $U2 := users$

$Answer := \Pi_{U1.uid,U2.uid,U1.name}(\sigma_{U1.uid>U2.uid \wedge U1.name=U2.name}(U1 \times U2))$

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.

   $MessagesToContact := Message \bowtie_{user=from \wedge contact=to} Contact$

   $NeverSent_(user, contact, first, last) := \Pi_{user,contact,0,0}Contact - (\Pi_{user,contact}User - \Pi_{user,contact}Mes$

   $NotFirst := \sigma_{M1.time>M2.time}[\rho_{M1}MessageToContact \bowtie \rho_{M2}MessageToContact]$

   $FirstMessage := MessagesToContact - NotFirst$

   $NotLast := \sigma_{M1.time<M2.time}[\rho_{M1}MessageToContact \bowtie \rho_{M2}MessageToContact]$

   $LastMessage := MessagesToContact - NotLast$

   $Answer_(user, contact, first, last) :=$
   $\quad \Pi_{F.from,F.to,F.time,L.time}[\rho_F FirstMessage \bowtie_{F.from=L.from \wedge F.to=L.to} \rho_L LastMessage]$

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 spammers 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).

   $S1 := Status$

   $S2 := Status$

   $M1 := Message$

   $M2 := Message$

   $$SpammerID := \Pi_{from}\sigma_{\substack{S1.status=delivered \wedge S1.mid=M1.mid \\ \wedge S2.status=read \wedge S2.mid=M2.mid \\ \wedge M1.time>M2.time}}(S1 \times S2 \times M1 \times M2)$$

   $\Pi_{from,lastSeen,photo,profile}(\sigma_{SpammerID.from=Privacy.uid}(SpammerID \times Privacy))$

## 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.

If a constraint cannot be expressed in relational algebra (with the operators we are considering in this assignment), simply write cannot be expressed.

1. The receiver (`Message[to]`) of a message must be either a user (`User[uid]`) or a group (`Group[gid]`)

   $\sigma_{to}(Message) - \sigma_{uid}(User) - \sigma_{gid}(Group) = \emptyset$

2. A user can only send messages to users in her contacts (`Contact[contact]`) and the time of the message must be after the start of the contact. This includes direct messages sent to a user and messages sent to a group. All members of the group must be in the users contacts.

   $G1 := Group$

   $G2 := Group$

   $C1 := Contact$

   $C2 := Contact$

   $GroupContacts := \Pi_{C1.user, C1.contact} \sigma_{\substack{G1.uid=C1.user \wedge g2.gid=g1.gid \\ \wedge g2.uid=C2.contact \wedge C2.user=C1.user}} (C1 \times C2 \times G1 \times G2)$

   $InvalidGroupMessage := \sigma_{to!=contact \vee time<start}(Message \bowtie_{from=user} GroupContacts)$

   $InvalidMessage := \sigma_{to!=contact \vee time<start}(Message \bowtie_{from=user} Contact)$

   $InvalidGroupMessage \times InvalidMessage = \emptyset$

3. The total size of all attachments in a message must be less than 128MB

   Cannot be expressed

4. The `Status` relation may not contain a message and user if the message was not sent to that user (either directly or the user was part of a group that received the message).

   $\sigma_{Status.uid!=to \vee Group.uid!=to}(Status \bowtie_{Status.mid=Message.mid} Message \bowtie_{to=gid} Group) = \emptyset$