University of Toronto
CSC343, Winter 2015

Assignment 1: Name:Ye Yuan Student Number:1001614247

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:

- $New(a,b,c) := R$

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 names such as "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. Report the user name of every student who has never worked with anyone, but has indeed submitted at least one file for at least one assignment.

   Answer:

— Student who has submitted at least one file for at least one assignment:
$Subatleast1(userName) := \Pi_{userName} Submission$

— Student who has worked with other students
$WorkedTogether(userName) := \Pi_{u1} \sigma_{u1 != u2 \wedge g1 = g2} (\rho_{T_1(u1,g1)} Membership \times \rho_{T_2(u2,g2)} Membership)$

— Student who has never worked with anyone
$WorkedAlone(userName) := \Pi_{userName} \sigma_{type='student'} User - WorkedTogether$

— final answer
$Answer(userName) := WorkedAlone \cap Subatleast1$

2. Find the graders who have marked every assignment. We will say that a grader has marked an assignment if they have given a grade on that assignment to at least one group, whether or not that grade has been released. Report the grader's userName.

   Answer:

   —If the grader has marked every assignment, he/she should appear in a table
   $ShouldHaveMarked(userName, aID) := \Pi_{userName} Grader \times \Pi_{aID} Assignment$

   —A grader's acctual marked relation can be shown like:
   $ActualMarked(userName, aID) := \Pi_{userName, aID} (Group \bowtie Grader \bowtie Result)$

   —The grader who has not marked every assignment
   $Missing(userName) := \Pi_{userName} (ShouldHaveMarked - ActualMarked)$

   —final answer
   $Answer(userName) := \Pi_{userName} Grader - Missing$

3. Find all groups for A2 (*i.e.*, the assignment whose description is "A2") whose last submission was after the due date, but who submitted at least two different files (*i.e.*, files with two different names) before the due date. Report the group ID, the name of the first file they submitted, and when they submitted it. If there are ties for a group's first submit, report them all.

   Answer:

   —If a group submitted assignment A2 once after due, we can certainly say that this group's last submission for assignment A2 is after due. So we don't need to check if this is the last submission.
   $AfterdueA2(gID) := \Pi_{gID} \sigma_{description='A2' \wedge when > due} (Submission \bowtie Assignment \bowtie Group)$

   —Group who also submitted at least two different files before due(since we select the same gID, the due must be the same, too. So when we compare when and due, it doesn't matter to choose d1 or d2)
   $Both(gID) := \Pi_{g1} \sigma_{g1 = g2 \wedge w1 < d1 \wedge w2 < d2 \wedge f1 != f2} (\rho_{T_1(g1,d1,f1,w1)} (\Pi_{gID, due, fileName, when}$

$AfterdueA2 \bowtie Group \bowtie Submission \bowtie Assignment) \times \rho_{T_2(g2,d2,f2,w2)}$

$(\Pi_{gID,due,fileName,when} AfterdueA2 \bowtie Group \bowtie Submission \bowtie Assignment))$

—These group's Not first submission

$NotFirst(gID, when) := \Pi_{g1,w2}\sigma_{g1=g2 \wedge w1<w2}(\rho_{T1(g1,w1)}(\Pi_{gID,when}(Both \bowtie Submission))$

$\times \rho_{T2(g2,w2)}(\Pi_{gID,when}(Both \bowtie Submission)))$

—Their first submission(if we their are ties in their first submission, we also need to include sid because otherwise we will get duplication in our solution(but the requirements doesn't allow to project sid) and ties will be eliminated in sets)

$FirstSubmission(gID, fileName, when) := \Pi_{gID,fileName,when}((\Pi_{gID,when}(Both \bowtie Submission) -$

$NotFirst) \bowtie Submission)$

4. Find pairs of students who worked in a group together, and without any other students, on each assignment in the database that allowed groups of size two or more. Report their user names, last names, and firstnames.

   Answer:

   —Assignment allow group size two or more:

   $AssignMaxgt2(aID, gID) := \Pi_{aID,gID}\sigma_{groupMax>=2}(Assignment \bowtie Group)$

   —group has at least two members

   $Atleast2(gID) := \Pi_{g1}\sigma_{u1!=u2 \wedge g1=g2}(\rho_{T1(u1,g1)}Membership \times \rho_{T2(u2,g2)}Membership)$

   —group has at least three members

   $Atleast3(gID) := \Pi_{g1}\sigma_{u1!=u2 \wedge u1!=u3 \wedge u2!=u3 \wedge g1=g2=g3}$

   $(\rho_{T1(u1,g1)}Membership \times \rho_{T2(u2,g2)}Membership\rho_{T3(u3,g3)}Membership)$

   —group has exactly two members

   $Exactly2(gID) := Atleast2 - Atleast3$

   —Group members in these group whose size is exactly 2

   $Exactly2user(user1, user2, gID) := \Pi_{T1.userName,T2.userName,T1.gID}$

   $\sigma_{T1.userName<T2.userName \wedge T1.gID=T2.gID}(\rho_{T1(gID,userName)}(Exactly2 \bowtie Membership) \times \rho_{T2(gID,userName)}$

   $(Exactly2 \bowtie Membership))$

   —Then we obtain their corresponding aID

   $Exactly2Assign(user1, user2, aID) := \Pi_{user1,user2,aID}(Exactly2user \bowtie Group)$

   —If these two student worked for all the assignment that allow group size of two or more,they should satisfy:

$Shouldhave(user1, user2, aID) := \Pi_{user1,user2} Exactly2user \times \Pi_{aID} AssignMaxgt2$

—Then we can check the missing part(which doesn't satisfy the condition 'each assignment'):

$Missing(user1, user2, aID) := Shouldhave - Exactly2Assign$

—Find all the required userName pairs

$Pairs(user1, user2) := \Pi_{user1,user2} Exactly2user - \Pi_{user1,user2} Missing$

—Then we can obtain the final answer

$Answer(user1, lastName1, firstName1, user2, lastName2, firstName2) :=$

$\Pi_{user1,T1.lastName,T1.firstName,user2,T2.lastName,T2.firstName} \sigma_{T1.userName=user1 \wedge T2.userName=user2}$

$(\rho_{T1} User \times Pairs \times \rho_{T2} User)$

5. Find any assignments where the highest mark given by one grader is less than the lowest mark given by another grader. In your result, include a row for each grader on each of these assignments. Report the assignment ID, the grader's userName, and their minimum and maximum grade.

Answer:

—Find the cases which are not one grader's lowest mark

$graderNotmin(aID, userName, mark) := \Pi_{a1,u1,m2} \sigma_{a1=a2 \wedge u1=u2 \wedge m1<m2}$

$(\rho_{T1(a1,u1,m1)} \Pi_{aID,userName,mark}(Grader \bowtie Group \bowtie Result) \times \rho_{T2(a2,u2,m2)} \Pi_{aID,userName,mark}$

$(Grader \bowtie Group \bowtie Result))$

—one grader's lowest mark for each assignment

$graderMin(aid, userName, markmin) := \Pi_{aID,userName,mark}(Grader \bowtie Group \bowtie Result) -$

$graderNotmin$

—Find the cases which are not one grader's highest mark

$graderNotmax(aID, userName, mark) := \Pi_{a1,u1,m1} \sigma_{a1=a2 \wedge u1=u2 \wedge m1<m2}$

$(\rho_{T1(a1,u1,m1)} \Pi_{aID,userName,mark}(Grader \bowtie Group \bowtie Result) \times \rho_{T2(a2,u2,m2)} \Pi_{aID,userName,mark}$

$(Grader \bowtie Group \bowtie Result))$

—one graders highest mark for each assignment

$graderMax(aID, userName, markmax) := \Pi_{aID,userName,mark}(Grader \bowtie Group \bowtie$

$Result) - graderNotmax$

—one graders highest and lowest mark for each assignment

$graderMinMax(aID, userName, markmin, markmax) := graderMin \bowtie graderMax$

—aID for assignments where the highest mark one grader given is less than the lowest mark another grader given

$graderMingtMax(aID) := \Pi_{a1}\sigma_{a1=a2 \wedge u1!=u2 \wedge min1>max2}(\rho_{T1(a1,u1,min1,max1)}graderMinMax$

$\times\ \rho_{T2(a2,u2,min2,max2)}graderMinMax)$

—project all the required attributes

$Answer(aID, userName, markmin, markmax) := graderMingtMax \bowtie graderMinMax$

6. Find all students who have worked in a group with at least one other person, but have never worked with the same person twice. Report their userName.

Answer:

—Find students who have worked in a group with at least one other person

$workedwithOther(user1, user2, gID) := \Pi_{u1,u2,g1}\sigma_{g1=g2 \wedge u1!=u2}(\rho_{T1(u1,g1)}Membership$

$\times\ \rho_{T2(u2,g2)}Membership)$

—Find students who worked with the same person twice(in the different group)

$workedwithSame(user1, user2, gId) := \Pi_{T1.u1,T1.u2,T1.g1}\sigma_{g1!=g2 \wedge T1.u1=T2.u1 \wedge T1.u2=T2.u2}$

$(\rho_{T1(u1,u2,g1)}workedwithOther \times \rho_{T2(u1,u2,g2)}workedwithOther)$

—final answer

$Answer(userName) := \Pi_{user1}workedwithOther - \Pi_{user1}workedwithSame$

7. Find all students who meet these two requirements: (a) their groups (whether they were working alone or with others) handed in every required file, and did so on time, for all assignments, and (b) their grades never went down from one assignment to another with a later due date. Report their userName. Note: If an assignment has no required files, it is true of any group that they handed in every required file.

Answer:

—All the assignment one student should submit

$Shouldhavesubmitted(userName, aID) := \Pi_{userName,aID}\sigma_{type='student'}User \times Assignment$

—The assignment one student actual submitted required file on time

$ActualSubmittedOntime(userName, aID) := \Pi_{userName,aID}$

$\sigma_{when<due}(\rho_{T1(userName,aID,when,fileName)}\Pi_{Membership.userName,aID,when,fileName})$

$\sigma_{Memership.gID=Submission.gID=Group.gID}(Membership \times$

$Submission \times Group) \bowtie \rho_{T2(due,aID,fileName)} \Pi_{due,aID,fileName}(Required \bowtie Assignment))$

—Find students who don't meet the requirements above

$Missing(userName) = \Pi_{userName}(Shouldhavesubmitted - ActualSubmittedOntime)$

—Find students who satisfy condition a

$Conditiona(userName) := \Pi_{userName}\sigma_{type='student'}User - Missing$

—Find students who satisfy condition a but their grader went down at least once

$Wentdown(userName) := \Pi_{T1.userName}\sigma_{T1.userName=T2.userName \wedge T1.due>T2.due \wedge T1.mark<T2.mark}$

$\rho_{T1}(Conditiona \bowtie Membership \bowtie Group \bowtie Assignment \bowtie Result) \times$

$\rho_{T2}(Conditiona \bowtie Membership \bowtie Group \bowtie Assignment \bowtie Result)$

—Find students that satisfy condition b and condition a(final solution)

$conditionab(userName) := Conditiona - Wentdown$

8. Find all assignments that have one or more groups with no grade or with a grade that has not been released. Report the assignment ID and description.

Answer:

—Report the gID whose mark doesn't show up in result

$Nograde(gID) := \Pi_{gID}Group - \Pi_{gID}Result$

—Report the groups whose mark is up but haven't released yet

$Norealsed(gID) := \Pi_{gID}\sigma_{released=false}Result$

—project these groups' aID and description

$Answer(aID, description) := \Pi_{aId,description}((Nograde \cup Norealsed) \bowtie Group \bowtie Assignment)$

9. Assignments may have required files, but students can also hand in other files that are not required. Find all groups that never handed in a file that was not required. Report the group ID.

Answer:

—Find all groups who has at least submitted not required file once

$SubmitNotrequired(gID) := \Pi_{Group.gID}$

$\sigma_{Group.gID=Submission.gID \wedge Group.aID=Required.aID \wedge Submission.fileName!=Required.fileName}$

$(Group \times Submission \times Required)$

—Find all groups who never submitted an unrequired file

$Answer(gID) := \Pi_{gID}Group - SubmitNotrequired$

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. No grades can be released for an assignment unless every group has been given a grade on that assignment (whether or not it has been released).

   Answer:

   —Find all the aID who still has at least one group not marked

   $Notgradedall(aID) = \Pi_{aID}((\Pi_{gID}Group - \Pi_{gID}Result) \bowtie Group)$

   —The group with such aID cannot be released

   $\sigma_{released=true}(Notgradedall \bowtie Group \bowtie Result) = \emptyset$

2. A TA can't give a grade to any groups on an assignment unless they have completed marking (*i.e.,* they have given a grade, whether or not it has been released) for every group they were assigned to grade on every assignment with an earlier due date.

   Answer:

   —Find all the TA.

   $TA(userName) := \Pi_{userName}\sigma_{type='TA'}User$

   —Find all the groups assigned to a TA but haven't been makred yet

   $NotMarked(gID, userName) := Grader \bowtie TA - \Pi_{gID,userName}(Grader \bowtie TA \bowtie Result)$

   —Find all the groups that a TA has marked

   $Marked(gID, userName) := \Pi_{gID,userName}(Grader \bowtie TA \bowtie Result)$

   —If there is one assignment unmarked but an assignment with later due is marked, it's not allowed

   $\sigma_{T1.userName=T2.userName \wedge T1.due<T2.due}\rho_{T1}(NotMarked \bowtie Group \bowtie Assignment) \times \rho_{T2}(Marked \bowtie Group \bowtie Assignment) = \emptyset$

3. A TA can't be assigned to grade a group of size two or more unless he or she has already given a grade (that has been released) to at least 3 students (each working in a group of size 1) on an assignment with an earlier due date.

Answer:

—Find all the TA.

$TA(userName) := \Pi_{userName}\sigma_{type='TA'}User$

—Find all the groups has two or more members

$Groupgt2(gID) := \Pi_{T1.gID}\sigma_{T1.gID=T2.gID \wedge T1.userName!=T2.userName}(\rho_{T1}Membership$

$\times \rho_{T2}Membership)$

—Find all the group has exactly one student

$Group1(gID) := \Pi_{gID}Group - Groupgt2$

—Find all TA who has already given a grade to at least 3 students(each working in a size of 1, Note that these 3 groups must belong to the same assignment)

$Group1experiencegt3(userName, due) := \Pi_{T1.userName,T1.due}$

$\sigma(T1.userName = T2.userName = T3.userName \wedge T1.gID! = T2.gID \wedge T1.gID$

$! = T3.gID \wedge T2.gID! = T3.gID \wedge T1.released = T2.released = T3.released = true$

$\wedge T1.aID = T2.aID = T3.aID)(\rho_{T1}(Grader \bowtie TA \bowtie Group1 \bowtie Result \bowtie Group$

$\bowtie Assignment) \times \rho_{T2}(Grader \bowtie TA \bowtie Group1 \bowtie Result \bowtie Group \bowtie Assignment)$

$\times \rho_{T3}(Grader \bowtie TA \bowtie Group1 \bowtie Result \bowtie Group \bowtie Assignment))$

—Find all the TAs who are assigned to grade a group of size two or more

$Assigned2ormore(userName, due) := \Pi_{userName,due}(TA \bowtie Grader \bowtie Groupgt2 \bowtie$

$Group \bowtie Assignment)$

—If one TA is assigned to mark a group of size two or more, he/she must have prior experience with 3 single-person group, which means that a grader's 1st experience of 3-single-person-group must earlier than the 1st 2-or-more group assignment. Hence, we need to find their 1st experience of grading at least 3 students.Let's first obtain the not first ones:

$NotfirstGroup1experiencegt3(userName, due) := \Pi_{T1.userName,T1.due}$

$\sigma_{T1.userName=T2.userName \wedge T1.due>T2.due}(\rho_{T1}Group1experiencegt3 \times \rho_{T2}Group1experiencegt3)$

—Then we obtain every TA's first 3 single-person-group experience's due of corresponding assignment

$FirstGroup1experiencegt3(userName, due) := Group1experiencegt3 -$

$NotfirstGroup1experiencegt3$

—Then we obtain one TA's Not 1st 2-or-more group assignment

$NotFirstAssigned2ormore(userName, due) := \Pi_{T1.userName,T1.due}$

$\sigma_{T1.userName=T2.userName \wedge T1.due>T2.due}(\rho_{T1} Assigned2ormore \times \rho_{T2} Assigned2ormore)$

— Find a TA's 1st 2-or-more group assignment
$FirstAssigned2ormore(userName, due) := Assigned2ormore - NotFirstAssigned2ormore$

—If one TA's FirstAssigned2ormore occured before his/her FirstGroup1experiencegt3, or one TA is Assigned2ormore but he/she doesn't has Group1experiencegt3 at all, it's not allowed, which means it's an emptyset.
$(\Pi_{T1.userName}\sigma_{T1.userName=T2.userName \wedge T1.due<T2.due}(\rho_{T1} FirstAssigned2ormore \times$

$\rho_{T2} FirstGroup1experiencegt3)) \cup (\Pi_{userName} FirstAssigned2ormore - \Pi_{userName}$

$FirstGroup1experiencegt3) = \emptyset$