University of Toronto
CSC343, Winter 2015

Assignment 1

Due: Tuesday 3 February at 8:00 pm sharp!

In this course, you will submit your assignments using the online tool MarkUs. MarkUs is "an open-source tool which recreates the ease and flexibility of grading assignments with pen on paper, within a web application. It also allows students and instructors to form groups, and collaborate on assignments." (reference: `http://markusproject.org/`). For this assignment, the domain you will be working on is itself MarkUs. You will write queries and integrity constraints on a (simplified) schema for the data that MarkUs stores.

Schema

Relations

- User(userName, lastName, firstName, type)
  A tuple in this relation represents a MarkUs user. *userName* is their cdf user name, *lastName* and *firstName* are obvious, and *type* is the kind of MarkUs user they are. (The possible values for *type* are defined in an integrity constraint below.)
- Assignment(aID, description, due, groupMin, groupMax)
  A tuple in this relation represents an assignment. *aID* is the assignment ID, *description* is a short description of the assignment, *due* is the date and time when it is due, *groupMin* is the minimum number of students who may work together on the assignment, and *groupMax* is the maximum number of students who may work together on the assignment.
- Required(aID, fileName)
  A tuple in this relation represents the fact that submitting a certain file is required for a certain assignment. *aID* is the assignment ID and *fileName* is the name of the required file.
- Group(gID, aID, repo)
  A tuple in this relation represents a group. *gID* is the group ID, *aID* is the assignment of the ID for which this group exists, and *repo* is the URL of the shared repository where their submitted files are stored.
- Membership(userName, gID)
  A tuple in this relation represents that a user belongs to a group. *userName* is their cdf user name, and *gID* is the group to which they belong.
- Submission(sID, fileName, userName, gID, when)
  A tuple in this relation represents the fact that a file was submitted. *sID* is the submission ID, *fileName* is the name of the file that was submitted, *userName* is the user name of the user who submitted the file, *gID* is the group ID of the group to which the assignment was submitted, *when* is the date and time when the file was submitted. The set of attributes {*fileName, userName, when*} is also a key.
- Grader(gID, userName)
  A tuple in this relation represents the fact that a user was assigned to grade a group. *gID* is the ID of the group and *userName* is the user name of the user who was assigned to grade them.
- Result(gID, mark, released)
  A tuple in this relation represents a mark for a group. *gID* is the group ID of the group who was given the mark, *mark*, a number, is the mark they got, and *released* is a boolean indicating whether or not the result has been made available for the group members to see on MarkUs.

Integrity constraints

- Required[aID] $\subseteq$ Assignment[aID]
- Group[aID] $\subseteq$ Assignment[aID]
- Membership[userName] $\subseteq$ User[userName]
- Membership[gID] $\subseteq$ Group[gID]
- Submission[userName] $\subseteq$ User[userName]
- Submission[gID] $\subseteq$ Group[gID]
- Grader[gID] $\subseteq$ Group[gID]
- Grader[userName] $\subseteq$ User[userName]
- Result[gID] $\subseteq$ Group[gID]
- $\Pi_{\text{type}}\text{User} \subseteq \{\text{"instructor"}, \text{"TA"}, \text{"student"}\}$
- $\Pi_{\text{type}}(\text{Membership} \bowtie \text{User}) \subseteq \{\text{"student"}\}$
- $\Pi_{\text{type}}(\text{Grader} \bowtie \text{User}) \subseteq \{\text{"TA"}, \text{"instructor"}\}$
- $\sigma_{\text{groupMin}<0\ \vee\ \text{groupMin}>\text{groupMax}}\text{Assignment} = \emptyset$
- **Correction:**
  $\sigma_{\text{G1}\neq G2}(\rho_{\text{M1(userName, G1, aID)}}(\text{Membership} \bowtie \Pi_{gID,aID}\text{Group})$
  $\bowtie \rho_{\text{M2(userName, G2, aID)}}(\text{Membership} \bowtie \Pi_{gID,aID}\text{Group})) = \emptyset$
- $(\Pi_{\text{userName,gID}}\text{Submission}) - (\Pi_{\text{userName,gID}}\text{Membership}) = \emptyset$

Warmup: Getting to know the schema

To get familiar with the schema, ask yourself questions like these (but don't hand in your answers):

- Can the same file be submitted more than once by the same group on the same assignment?
- Are group IDs unique across each individual assignment or across all assignments?
- How many different graders can mark an assignment? How many can mark a single group's work on an assignment?
- What does each integrity constraint mean in plain English?

Part 1: Queries

Write the queries below in relational algebra. There are a number of variations on relational algebra, and different notations for the operations. You must use the same notation as we have used in class and on the slides. You may use assignment, and the operators we have used in class: $\Pi, \sigma, \bowtie, \bowtie_{condition}, \times, \cap, \cup, -, \rho$. Assume that all relations are sets (not bags), as we have done in class.

Remember that the condition on a select operation may only examine the values of the attributes in one tuple (not whole columns), and that it can use comparison operators (such as $\leq$ and $\neq$) and boolean operators ($\vee$, $\wedge$ and $\neg$). One relation in our schema has a date-time attribute. You may use comparison operators on such values.

You are encouraged to use assignment to define intermediate results, and it's a good idea to add commentary explaining what you're doing. This way, even if your final answer is not completely correct, you may receive part marks.

Do not make any assumptions about the data that are not enforced by the original constraints given in the schema. Your queries should work for any database that satisfies those constraints.

Note: These queries are not in order according to difficulty.

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.
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.
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.
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.
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.
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.
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.
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.
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.

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).

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.
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.

When writing your queries for Part 1, don't assume that these additional integrity constraints hold.

Style and formatting requirements

In order to make your algebra more readable, and to minimize errors, we are including these style and formatting requirements:

- In your assignment statements, you must include names for all attributes in the intermediate relation you are defining. For example, write

$$HighestGrade(gID, aID, grade) := \quad \ldots$$

- Use meaningful names for intermediate relations and attributes, just as you would in a program.
- If you want to include comments, put them before the algebra that they pertain to, not after. Make them stand out from the algebra, for example by using a different font.

A modest portion of your mark will be for good style and formatting.

Submission instructions

Your assignment must be typed; handwritten assignments will not be marked. You may use any word-processing software you like. Many academics use LaTeX. It produces beautifully typeset text and handles mathematical notation well. If you would like to learn LaTeX, there are helpful resources online. Whatever you choose to use, you need to produce a final document in pdf format.

You must declare your team (whether it is a team of one or two students) and hand in your work electronically using the MarkUs online system. Instructions for doing so are posted on the Assignments page of the course website. Well before the due date, you should declare your team and try submitting with MarkUs. You can submit an empty file as a placeholder, and then submit a new version of the file later (before the deadline, of course); look in the "Replace" column.

For this assignment, hand in just one file: A1.pdf. If you are working in a pair, only one of you should hand it in.

Check that you have submitted the correct version of your file by downloading it from MarkUs; new files will not be accepted after the due date.