University of Toronto
CSC343, Fall 2021

Sample Relational Schema

This handout contains an example of a schema that is more complex that what you've seen so far. It is provided as an example of good schema design and good documentation, which you can use as inspiration for your Project schema. But note that your project schema will likely be simpler than this one!

You can also use this example to practice reading and interpreting a relational schema. For example, ask yourself questions like these: According to the schema,

- Can the same file be submitted more than once by the same group on the same assignment?
- Can a file be submitted for an assignment after the due date?
- 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?

Bonus: You are getting used to the schema that you will be working with on Assignment 1.

Schema

This schema is for the online tool MarkUs, which you will use to submit your work in this course. 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/`).

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 or other course item that is submitted on MarkUs, such as a test, project, or quiz. *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($\underline{\text{userName, gID}}$, status)
  A tuple in this relation represents that a user belongs to a group. *userName* is their cdf user name, *gID* is the group to which they belong, and *status* records information about which member invited which to the group and whether an invitation was accepted (see the constraints below for more information).

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