Abusing grading schemes: Almost anything can be encoded in a grading scheme

Gerald Q. Maguire Jr.

Some time ago I mused about administrative assignments to be able to keep track of information concerning a student. At the time I turned to custom columns as these can have arbitrary text in them (but the field is limited in length). However, one of the perils of custom columns is that while the values are written to the database - there is **no** API to be able to recover previous values.

So this brought me back to thinking about an assignments. Well, it turns out that you can put arbitrary strings as the "name" of a level in a "letter grade" grading scheme. One of the values I would like to keep track of is who the examiner is for a given student when they do their degree project (such as a Bachelor's or Master's thesis). It turns out that you can put the examiners' names into a grading scheme and then add a value field to each of them.

# Initial state

After defining an assignment (“Select examiner” - see Section 5) with the new grading scale (see Section 4). Figure 1 shows the initial state of the gradebook for the Test Student (see the field on the right).

|  |
| --- |
|  |

Figure 1: Initial state of the gradebook with a new assignment “Select examiner”

# Set a "grade", i.e., select the examiner's name

The user clicks on the cell as usual to assign a “grade” and they get a list of the examiners’ names, as shown in Figure 2.

|  |
| --- |
|  |

Figure 2: Selecting an examiner from the list

# State in the gradebook

This updates the “grade” in the gradebook as shown in Figure 2.

|  |
| --- |
|  |

Figure : State of gradebook after selecting the examiner

# What does such a "grading scale" look like?

The grading scaled (named II246X) used for the above is shown in Figure 4. This can also be seen for another grading scheme shown in Figure 5.

|  |
| --- |
|  |

Figure : II246X grading scheme

|  |
| --- |
|  |

Figure : Grading scheme – another view

# The “Select examiner” assignment

The “Select examiner” assignment is shown in Figure 6.

|  |
| --- |
|  |

Figure : Select examiner assignment

# Limitations

Is there a limit to the number of "grade levels"? I should check the source code to see if there is an upper limit. However, my quick tests showed it worked for at least 50 names.

Note that the program uses the information about courses and examiners as extracted from KOPPS. The source code can be found at insert\_teachers\_grading\_standards.py at <https://github.com/gqmaguirejr/E-learning> .

I am not so sure that listing the examiners ordered by their first name is the best way to list the examiners in a grading scheme, but it was easy to do. In my earlier LTI based method of getting this data from a pulldown list in a web page, generated by ruby, I used Canvas' sort-names, i.e., ordered by their last name.

Could it be better, yes - it would be nice to be able to put the examiner's Canvas user\_id in, rather than their name - while still having their name shown in the list. However, this would mean changes would be needed to the code that displays the gradebook. In these approach described in this document, there are no changes to Canvas required.

# To be done

The current code assigns a “value” for exam examiner based upon their order in the list and the total number of examiners in the list of examiners for a given course.

The next task is to figure out how to take the information for all of the examiners in the department or school and assign each of them a "value" to have the same value used for a given examiner across courses. If done cleverly this should enable the addition of new examiners, without having to renumber all of the existing examiners (after all the values are floating-point numbers, so there should be plenty of room between them 😀).

In this way only the examiners for a given course would be in the grading scheme for use with that course, but their numeric value would be the same in each course.

# Comparison with LTI and ruby code for storing examiner information

As noted earlier the Ruby code uses a custom column to store the text of the examiner’s name in the gradebook. This has the weakness of the custom columns being in the database, but earlier values are not accessible via the Canvas API. In contrast using an assignment and encoding the examiner information via the grading scheme it is possible to use the gradebook history API to see who assigned the values and when they did so (as well as what value was assigned).

Because the Ruby code uses the same information about courses and examiners as used here, the information is the same. However, in the case of the Ruby program the code could check which course the student was in an ensure that the list of examiners to choose from were only those for the specific course code. In contrast, using a grading scheme means that different quizzes/assignments need to be used for each different grading scale; thus if there are several course codes all in the same Canvas course, you would have separate assignments for each of the course codes – which in turn would each have its own grading scheme. Thus leading to potentially a number of different assignments (Note that these assignments can be made to specific sections within the course – so that different sections had different assignments/quizzes each with their own grading scheme.).

Probably the best scheme is to use the LTI approach combined with a grading scheme with all of the examiners in it-the Ruby code could ensure that the assignment of an examiner was compatible with the course code and program of the student. In this way one could have a single column in the gradebook for this information for students in all sections of the Canvas course.