

Name: _____

Overview: The main purpose of this assignment is to let people who did not get a good score on earlier assignments have a chance to improve their overall score. To that end, this assignment is *special*, in the following sense. If you do not submit this assignment, the 15% of your course grade that depends on homework assignments will be computed as $g_1 = 15 \times (s_1 + s_2 + s_3)/250$, where s_i is your score on assignment i . (Note that the first assignment is worth 50 points while the others are worth 100.) If you do submit this assignment, your score grade for the assignments will be computed as $\max(g_1, g_2)$, where $g_2 = 15 \times (s_1 + s_2 + s_3 + s_4)/350$.

This assignment is a simple extension to the previous assignment. (Thus, in addition to what is described here, everything described in the previous assignment must work properly in your submission.) If you have successfully finished the earlier assignments, this one should not take you very long to complete. For implementing the functions described below, follow the framework used in PHW03 (function page, result page, etc.). The additional functions should be included in your application home page.

function `updateStatus(foo, bar)` As a simple extension to the application in PHW03, add a function `updateStatus`, which takes two arguments. The first is the SID of a Student-Book tuple and the second is a string denoting a loan status (e.g., “returned”). Status should be an extra field of the BookLoans table. Whenever a tuple is inserted or altered, the corresponding tuple status should be set to “loaned”. The only way to set status to “returned” is by using `updateStatus(foo, bar)`. The SID (`foo`) will be a value previously output by your application. (See the discussion on SIDs in PHW03 for details.) If the given SID value is not valid (e.g., it does not correspond to a tuple in the BookLoans table) an error should be flagged using the diagnostics area of the results page (as described in PHW03). If the SID is valid, the action code of the corresponding tuple in BookLoans should be updated to the given string (`bar`). This function produces no output. You can presume that the values used to set a tuple’s status will be limited to “loaned” and “returned”. You should implement this as a drop-down menu with these options in the `updateStatus` function’s html page.

Audit Trail Your main task is to add an *audit trail* feature to the application you built in the last assignment. An audit trail is, informally, a record of modifications made to the database (including the type of the modification, the date and time at which it occurred, etc.). For this assignment, we wish to keep track of all changes made to the BookLoans table. For example, if a tuple is inserted, we wish to record the inserted tuple and a timestamp denoting the time and date of insertion. If a tuple is updated using the function `updateStatus` described above, we wish to record the old and new values of the action code along with the

timestamp of modification. For a deleted tuple, we wish to record the values of all attributes of the tuple along with the timestamp of the deletion. Note that the audit trail should be persistent. That is, an audit trail created and used in a session with your application today should be visible (using `showAuditTrail` described below) in another session, say, tomorrow, unless it has been destroyed at the user's request.

Important: You must implement this assignment in a manner that produces a proper audit trail for all modifications made to the `Shipment` table, even if they are made by bypassing your application (e.g., using `sqlplus`).

As part of this audit trail feature, you must implement the following functions:

Function `createAuditTrail()` This function should create a new audit trail, initialized to empty. (Any existing audit trail should be deleted or otherwise made to disappear from the user's view.) This function produces no output.

Function `destroyAuditTrail()` This function should result in the current audit trail (if it exists) being destroyed. It is not an error to invoke this function when there is no audit trail. This function produces no output.

Function `resetAuditTrail()` This function should empty the audit trail of all previously stored transactions, but *not* delete it. This function produces no output.

Function `showAuditTrail()` This function should display the current state of the audit trail. For each modification represented in the audit trail, it should output a record consisting of the timestamp and the details of the modification. If the modification is an insertion or deletion, the details should include the values of all attributes in the inserted or deleted tuple. If it is an update, the details should include the old and new values of the updated attribute along with the values of the other attributes. The values should represent the state of the database at the time of modification which, in general, will be different from the current state. For example, if a tuple is inserted and then updated, the details part of the audit trail record for the insertion should include the original values of the inserted tuple, not the values after the subsequent update operation. The trail timestamp must be output using the default format used by the Unix `date` program (e.g., `Thu Nov 21 09:01:07 EST 2002`). You can format the details in a manner of your choosing, as long as it is easy to figure out the items mentioned above. (Be sure to indicate the correspondence between the values you output and the attribute names, perhaps using a table.) This function will only be invoked when the audit trail is required to exist.

Function `showTodaysTransactions()` This function should display all the modifications applied to the `BookLoans` table that have a timestamp after the last midnight. The format of the modifications displayed should be the same as in `showAuditTrail()`.

Function `showLastNTransactions(n)` This function should display the last n transactions in the audit trail. If n is larger than the number of recorded modifications, `showLastNTransactions` should return output identical to `showAuditTrail`, i.e. *all* modifications that are recorded in the audit trail. The format of the output should follow the output of `showAuditTrail` function.

Reminders You are to implement the functionality described in this assignment *in addition* to that described in PHW03. Thus, everything described in PHW03 must work properly (including proper starting and stopping of the HTTP server, HTML validation, and the other functions). Please be careful when testing your code. In particular, please make sure you do not leave httpd processes running on the class machines.

Submission Follow the submission instructions for PHW03, replacing phw03 with phw04 appropriately. Please make sure that your submission can be uncompressed, unpacked, compiled, and executed properly. Test it carefully! There is no need to submit any hardcopy in class. We will grade the most recent file you upload to the FTP server before the deadline.