CS 328 - Homework 4

Deadline

Due by 11:59 pm on **Sunday, February 19, 2017**

Purpose

To practice some more with PL/SQL stored subroutines and "strict"-style HTML5 (now also including forms)

How to submit

Submit your files for this homework using `~st10/328submit` on nrs-projects, with a hw number of `4`.

Important notes

- Note: you *may* be presenting versions of some of these HTML5 pages to the class at some point.
- Remember to follow the **CS 328 SQL and PL/SQL Coding Standards** as given in the CS 328 Homework 2 handout and the public course web site for all SQL and PL/SQL code.
- Make sure that you have executed the scripts `create-bks.sql` and `pop-bks.sql`, and that the bookstore tables are successfully created and populated.
- Remember to follow the **CS 328 HTML5 Coding Standards** as discussed in class and as given on the public course web site for all HTML5 documents. (And remember that these include the requirement that, to receive full credit, all of your HTML5 documents must validate as strict-style HTML5.)

Problem 1

CAVEAT: Yes, I realize that a sequence or some other means of auto-numbering *would* be preferred here. This problem's purpose is give you more practice writing PL/SQL subroutines that **call other** PL/SQL subroutines.

Create a file `328hw4.sql`.

(Make sure that you have a copy of `pop-bks.sql` in the same directory as `328hw4.sql` -- it is called in this problem's testing script, to make sure the tests are run on "fresh", original versions of these tables, before the tests muck with them. Note that this script happens to already end with a `commit;` command.)

Start your `328hw4.sql` file with the following:

- comments containing at least your name, `CS 328 - Homework 4`, and the last-modified date
- include the command to `set serveroutput on`
- followed by a SQL*Plus `spool` command to spool the results of running this SQL script to a file named `328hw4-out.txt`
- followed by a `prompt` command including your name

Be sure to `spool off` at the end of this script (after your statements for the remaining problems).

Then, write a SQL*Plus `prompt` command that says `problem 1`. (You may add additional `prompt`

commands around this to make it more visible, if you would like.)

Remember `next_ord_needed_id` from Homework 3, Problem 1? You are now going to write a PL/SQL stored procedure calls that stored function. (Remember, `next_ord_needed_id` is **already stored** in your database -- you can simply call it from this script, you need **not** recreate it.)

When we want to add rows to table `order_needed`, we will call a PL/SQL stored procedure `insert_order_needed` whose parameters are the desired ISBN and order quantity for the desired order. This will do the following:

- insert a new row into `order_needed` using a key returned by calling `next_ord_needed_id`, the parameter ISBN and parameter order quantity, and the current date. (It should **NOT** fill the `date_placed` attribute for `order_needed`; that attribute should have the value null.)
- (since this order is needed, but not yet placed, you should NOT do anything to the `title`'s `on_order` attribute yet -- when an order is actually placed, THAT's when the `title`'s `on_order` attribute should be updated. We'll address that in another homework.)

Separately (outside of `328hw4.sql`) run the posted testing script `insert_order_needed_test.sql` posted along with this homework handout, make sure the tests all pass, and submit the file it spools with the test results, `insert_order_needed_test_out.txt`, along with your files for this homework.

You may also add additional testing calls along with your `insert_order_needed` code in your `328hw4.sql` script if you would like.

Problem 2

In `328hw4.sql`, write a SQL*Plus `prompt` command that says `problem 2`. (You may add additional prompt commands around this to make it more visible, if you would like.)

Consider again the `order_needed` table in the bookstore database. The idea/hope here is that, when a title's quantity on hand becomes less than the `order_point`, then a row should be added to the `order_needed` table indicating that, well, an order is needed for that title. Recall -- as noted above -- that, initially, the `date_placed` attribute for that new `order_needed` row is null -- the order is needed, but it has not yet been placed.

When, later, the order IS placed, then the `date_placed` attribute for the corresponding `order_needed` row is filled accordingly. (I hope to have you write a trigger on an upcoming homework that will take care of this.)

At any given time, then, the rows in `order_needed` that have a null value for `date_placed` indicate orders that, well, need to be made. One could think of these as "pending" `order_needed` rows.

There could be times (indeed, as part of an application you will be working on later this semester) where you would like to know if, for a given ISBN, there is a "pending" `order_needed` row for that ISBN (if there is a row with that ISBN whose `date_placed` attribute is null).

Write this little PL/SQL stored function `pending_order_needed` that expects an ISBN, and returns a `boolean` value indicating whether or not that ISBN has such a "pending" `order_needed` row.

Separately (outside of `328hw4.sql`) run the posted testing script `pending_order_needed_test.sql` posted along with this homework handout (note that it uses the stored function `bool_to_string` which was created by one of Homework 3's testing SQL scripts, `is_on_order_test.sql`, so it should still be in your database). Make sure the tests all pass, and submit the file it spools with the test results, `pending_order_needed_test_out.txt`, along with your files for this homework.

You may also add additional testing calls along with your `pending_order_needed` code in your

`328hw4.sql` script if you would like.

Make sure that you submit your `328hw4.sql` and `328hw4-out.txt` files as well as the `*_test_out.txt` files for each of the subroutines in Problems 1 and 2.

Problem 3

Consider your `bks-start.html` from Homework 3, Problem 5.

Make a **new** copy of this page in a **different** directory, since you will be modifying it and you don't want to change Homework 3's version of this (since that could affect your Homework 3 grade!)

Add a link from your `index.html` on nrs-projects to this **new** version of `bks-start.html` (making clear in some OBVIOUS fashion this is **Homework 4's** version, and **NOT** removing the earlier link to Homework 3's version!!) **It will be an automatic -10 penalty on this homework if this link does not display this page when I/the grader attempt to click on it in a browser.**

Eventually, it is going to be handy for a qualified user to log in from here. So, add a form that allows the user to enter and then submit their Oracle username and password:

- your form's action can be the URL of any functioning web page -- later, we'll replace this with a URL that will actually attempt to process this form.
- give your form a `method` attribute with a value of `"post"` (although while you are debugging you can use `"get"`, as long as you replace it with `"post"` for the version that you submit)
- for the password entry, use a **password field** instead of a textfield (this is an `input` element with `type="password"` -- it is VERY much like a textfield, except when a user types into it, little circles or asterisks are displayed instead of what is actually typed.

  BUT NOTE!!!!! that it does NOT obscure the actual password submitted in any way -- you still need `https` to encrypt it en route, and if your form uses `method="get"`, the password typed in WILL be displayed at the end of the `action` attribute's URL in plain text...!)
- don't forget a submit button, an `input` element with `type="submit"`! (**NOTE** that including one of those in **every** `form` element whose contents are to be submitted is one of the CS 328 course coding standards.)

On a later homework, you'll format/layout this document and this form using CSS3. So, just concentrate on getting the appropriate form controls in here for this problem. But make sure your resulting page validates as strict-style HTML5.

Problem 4

Consider Problem 1's PL/SQL stored procedure `insert_order_needed`. What might an HTML5 form look like that could submit the information needed by this procedure to the application tier, so that the application tier could call this stored procedure (in conjunction with the data tier)?

Create an HTML5 document `insert-ord-needed.html` containing such a form that meets the following specifications:

- Include the URL from which it can be run within a comment near its beginning. **It will be an automatic -10 penalty on this homework if this URL does not display this page when I/the grader attempt to use it in a browser.**
- Add form controls for all of the information that a call to PL/SQL stored procedure

`insert_order_needed` expects.

  - For some of the information `insert_order_needed` expects, an `input` element with `type="text"` -- a textfield -- will be fine.
  - But, requiring the user to type in the desired ISBN is not very pleasant for the user. So, read about the `select` element -- HTML5's drop-down menu -- in Section 6.2.4, on pp. 217-219, of the course text, and then include a `select` element in your form for allowing the user to select a desired ISBN instead.
    - Later in the semester, we'll actually populate this `select` element on the application tier, querying for the ISBNs and looping through the results to build this element.
    - For now, use a `select` element for this in your form that you hard-code (type in specifically by hand) with just 4-5 of the bookstore database's ISBNs so you can get an idea of how this element works without having to hard-code all of the ISBNs currently in the bookstore database!

- This form also needs to allow the user to enter their Oracle username and password. For the password entry, use a password field (an `input` element with `type="password"`) rather than a textfield (as described in Problem 3).
- This HTML5 document should somehow visibly include, in its `body` element, your name, CS 328, and the name of your bookstore from `bks-start.html` .
- Your form's action can be the URL of any functioning web page -- **later**, we'll replace this with a URL that will actually attempt to process this form, building a call to the PL/SQL stored procedure `insert_order_needed` .
- Your form's method should be `"post"` (although while you are debugging you can use `"get"`, as long as you replace it with `"post"` for the version that you submit).
- Do **NOT** add a link from your `index.html` on nrs-projects to your resulting `insert-ord-needed.html` !!

On a later homework, you'll format/layout this document and this form using CSS3. So, just concentrate on getting the appropriate form controls in here for this problem.

Problem 5

To give you a chance to practice with some additional HTML5 elements (especially form control elements)...

...using the posted `html5-template.html` as the initial basis, create an opening "strict" HTML5 document `hw4-play.html` that simply permits you to practice with some of these. It should include at least:

- the URL from which it can be run included within a comment near its beginning. **It will be an automatic -10 penalty on this homework if this URL does not display this page when I/the grader attempt to use it in a browser.**
- your name visibly included somewhere within its `body` element.
- CS 328 visibly included somewhere within its `body` element
- an `h1` element of your choice
- a `table` element with at least 3 rows and 3 columns, with content of your choice
  - you can read about this element in the course text in Section 2.2.2 on pp. 33-35
  - as noted in the class HTML5 style standards, " you are expected to include a `caption` element within

each `table` element describing the table (to help with accessibility).", and

  - "It is also a course style standard to include a `scope` attribute for `th` elements, indicating if that table header cell is the header of a row (`scope="row"`) or of a column (`scope="col"`)."

- a form, whose subject(s) is/are your choice, that includes at least the following controls:
  - a submit button (of course) (an `input` element with `type="submit"`)
  - at least 3 checkboxes
    - you can read about this element in the course text in Section 6.2.2 on pp. 213-214
    - NOTE for **FUTURE** Exam 1 use: what name=value pairs do these submit if they are selected when the form is submitted?
  - at least 3 grouped radio buttons
    - you can read about this element in the course text in Section 6.2.2 on pp. 214-215
    - make sure these have the **same** value for their `name` attribute,
    - and make sure these have **different** values for their `value` attribute!
    - explicitly use `checked="checked"` to explicitly indicate which radio button should be checked initially when the document appears
    - NOTE for **FUTURE** Exam 1 use: what name=value pair is submitted for the selected radio button when the form is submitted?
  - at least one drop-down menu/box
    - you already read about this element in the course text in Section 6.2.4 on pp. 217-219, and used it in Problem 4's form for allowing the user to select an ISBN.
    - NOTE for **FUTURE** Exam 1 use: what name=value pair is submitted for the selected choice when the form is submitted?
  - at least one textfield (an `input` element with `type="text"`)
  - at least one `textarea` element
    - you can read about this element in the course text in Section 6.2.1 on pp. 211-212 (its discussion starts near the very end of p. 211)
    - NOTE for **FUTURE** Exam 1 use: what name=value pair is submitted for this when the form is submitted?
  - your form's action can be the URL of any functioning web page
  - give your form a `method` attribute with a value of `"post"` (although while you are debugging you can use `"get"`, as long as you replace it with `"post"` for the version that you submit)
  - (you may also add additional element(s) from the course textbook that you are interested in trying out)
- Do **NOT** add a link from your `index.html` on nrs-projects to your resulting `hw4-play.html`

On a later homework, you'll format/layout this document and this form using CSS3. So, just concentrate on getting the appropriate form controls in here for this problem.

Problem 6

Consider the PL/SQL stored procedure or stored function that you wrote for "your" database for Homework 2 - Problem 7.

What might an HTML5 form look like that could submit the information needed by your PL/SQL stored procedure or stored function to the application tier, so that the application tier could call this stored procedure or stored function (in conjunction with the data tier)?

Create an HTML5 document `custom-plsql.html` containing such a form that meets the following specifications:

- Include the URL from which it can be run within a comment near its beginning. **It will be an automatic -10 penalty on this homework if this URL does not display this page when I/the grader attempt to use it in a browser.**
- Add appropriate form controls for all of the information that a call to your PL/SQL stored procedure or stored function expects.
- This form also needs to allow the user to enter their Oracle username and password. For the password entry, use a password field (an `input` element with `type="password"`) rather than a textfield (as described in Problem 3).
- This HTML5 document should somehow visibly include, in its `body` element, your name, CS 328, and the name of your scenario from your `custom-start.html` .
- Your form's action can be the URL of any functioning web page -- **later**, we'll replace this with a URL that will actually attempt to process this form, building a call to your PL/SQL stored procedure or stored function.
- Your form's method should be `"post"` (although while you are debugging you can use `"get"`, as long as you replace it with `"post"` for the version that you submit).
- Do **NOT** add a link from your `index.html` on nrs-projects to your resulting `custom-plsql.html` !!

On a later homework, you'll format/layout this document and this form using CSS3. So, just concentrate on getting the appropriate form controls in here for this problem.

(NOTE: if for any reason you want to change or replace your PL/SQL stored procedure or stored function from Homework 2 - Problem 7, you may, BUT you must:

- still follow the specifications from Homework 2 - Problem 7 in your revised/new subroutine
- submit your new version in a file `328hw2-7-revised.sql` (and also submit its accompanying `328hw2-7-revised-out.txt`) along with your `custom-plsql.html` so I will know you have revised/replaced it.)