CS 318 - Homework 2

Deadline:

Due by 11:59 pm on Wednesday, February 13, 2013

How to submit:

Submit your files for this homework using `~st10/318submit` on nrs-projects, with a homework number of `2`

Instructions for using the tool ~st10/318submit:

- If they are not already on nrs-projects, transfer your files to be submitted to a directory on nrs-projects.
  - You can do so by using `sftp` (secure file transfer) and connecting to `nrs-projects.humboldt.edu`, and then transferring them
- Once all of your files to be submitted are in a directory on nrs-projects, then use `ssh` (or Putty in an Academic Computing lab) to connect to `nrs-projects.humboldt.edu` .
- use `cd` to change to the directory containing the files to be submitted -- for example,

  ```
  cd 318hw02
  ```

- type the command:

  ```
  ~st10/318submit
  ```

  ...and give the number of the homework being submitted (or whatever number you have been asked to do for lab-related files) when asked, and answer that, `y`, you do want to submit all of the files with of-interest-to-318 suffixes in the current directory. (Note that I don't mind if a few extraneous files get submitted as well -- I'd rather receive too many files that too few, and typing in all of the file names for each assignment is just too error-prone...)
- you are expected to carefully check the list of files that the tool believes have been submitted, and make sure all of the files you hoped to submit were indeed submitted! (The most common error is to try to run `~st10/318submit` while in a different directory than where your files are...)

Purpose:

To practice some more with PL/SQL stored procedures and functions (including calling some previously-written stored functions), and to practice with "strict" HTML5.

Important notes:

- Remember to follow the **CS 318 SQL and PL/SQL Style Standards** as given in the CS 318 Homework 1 handout 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.
- Unless explicitly indicated otherwise, for the entire semester, all web pages submitted are expected to use "strict" style HTML5, as discussed in class and in the course textbook.
- Likewise, unless explicitly indicated otherwise, all web pages submitted are expected to include the link to the W3C experimental HTML5 validator as well as the link to http://lint.brihten.com/html/ as shown in example page `html5-template.html`, and all must validate/pass the tests from both. Each page that does not will cause a loss of points on the problem involved.
- I'm not requiring specific indentation for HTML5 yet - I reserve the right to do so, however, if necessary. In the meantime, find a readable way of indenting it, and consistently do so...

The Problems:

Problem 1:

CAVEAT: Yes, I realize that a sequence *would* be preferred here. This problem's purpose is give you more practice writing PL/SQL subroutines that use other PL/SQL subroutines.

- Create a file `318hw2.sql` . Start this file with comments containing at least your name, `CS 318 - Homework 2`, and the last-modified date.
- (Make sure that you have a copy of `pop-bks.sql` in the same directory as `318hw2.sql` -- it is called below in the testing section, 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.)
- 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 `318hw2-out.txt` . 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_on_key` from Homework 1, Problem 6? You are now going to write a PL/SQL stored procedure that makes use of that stored function. (Remember, `next_on_key` 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 `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_on_key`, the parameter ISBN and 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.)
- After this stored procedure, execute the following testing code:

```
prompt
prompt ********************************
prompt TESTING insert_order_needed
prompt ********************************
prompt

start pop-bks

update title
set    qty_on_hand = qty_on_hand - 5
where  isbn = '0805322272';

exec insert_order_needed('0805322272', 50)

prompt
prompt ============================================================
prompt test passes if there is now an order_needed row for
prompt     0805322272, for 50 copies, with date_created of today,
prompt     and date_placed that is empty/null
prompt ============================================================
prompt

select *
from   order_needed;

prompt undoing temporary testing changes

rollback;
```

- You may add additional testing calls if you would like (placed before the `rollback;` command, of course).

Problem 2:

- In `318hw2.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.
- To test this, we'll again use Homework 1's testing stored function from Homework 1, Problem 7, `bool_to_string`. (This is already stored in your database, so you can simply call it from this script.)
- Follow your function `pending_order_needed` with the following testing code:

```
prompt
prompt **********************
prompt TESTING pending_order_needed
prompt **********************
prompt

update title
set qty_on_hand = qty_on_hand - 5
where isbn = '0805322272';

exec insert_order_needed('0805322272', 50)


prompt ============================================================
prompt test passes if returns true (IS a pending order_needed for
prompt     0805322272)
prompt ============================================================
prompt
var status_str varchar2(5);
exec :status_str := bool_to_string(pending_order_needed('0805322272'))
print status_str

prompt
prompt ============================================================
prompt test passes if returns false (order_needed row for
prompt     025602796X is NOT pending)
prompt ============================================================
prompt
exec :status_str := bool_to_string(pending_order_needed('025602796X'))
print status_str

prompt
prompt ============================================================
prompt test passes if returns false (0131103628 is a title,
```

```
prompt     but is not in order_needed table at all, so can't
prompt     be pending)
prompt ============================================================
prompt
exec :status_str := bool_to_string(pending_order_needed('0131103628'))
print status_str

prompt
prompt ============================================================
prompt test passes if returns false (1111111111 is NOT a title,
prompt     so can't be pending)
prompt ============================================================
prompt
exec :status_str := bool_to_string(pending_order_needed('1111111111'))
print status_str

prompt undoing temporary testing changes

rollback;
```

- You may add additional testing calls if you would like (again, before the `rollback;` command).
- Now turn spooling off. The resulting `318hw2.sql` and `318hw2-out.txt` files should now be ready to submit. (Although I have no problem with you submitting incomplete versions of these early on if, for example, you want to submit after you have completed problem 1... 8-))

Problem 3:

Along with this handout, you'll find the content for a file `hw2-warmup-before.html`. Currently, however, this content is NOT valid "strict" HTML5.

- Use this content as the initial content of a file `hw2-warmup-after.html`
- Add a level-2 heading that indicates that you modified this page (including your name)
- Correct this content to make it valid "strict" HTML5 (that successfully validates as HTML5 using the W3C experimental HTML5 validator AND passes the HTML lint tests at http://lint.brihten.com/html/, using the links included at the bottom of the page).
- Add an HTML comment that contains the proper URL to now be able to display your completed version of this page in a web browser.
  - Note that, to receive full credit for this problem, if I paste the URL you give in that comment into a browser, it must lead to your `hw2-warmup-after.html` that validates as valid "strict" HTML5 (using both linked tools) and meets the above specifications.
- The resulting `hw2-warmup-after.html` should be ready to submit.

Problem 4:

Consider the database created and populated using `create-bks.sql` and `pop-bks.sql`. You are going to gradually add some database application pieces atop this database over the course of the semester.

To start: create an opening "strict" HTML5 "title page" `bks-splash.html` that includes at least:

- your name
- CS 318
- a name of your choice for this bookstore
- an image of your choice
- a link (with appropriate descriptive text) leading to your file `hw2-warmup-after.html` from Problem 3
  - we'll remove this later; this is just to give you practice writing HTML for a hypertext link
- a form that allows the user to enter and then submit his/her 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, building and requesting that the database server perform some appropriate query
  - 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` tag with a `type` attribute value of `"password"`.

    (Its possible attributes and behavior and look is very like a textfield, except what the user types is obscured, so that someone looking over a user's shoulder can't read something sensitive, such as a password, being typed into it.)
  - we'll "format" this form using CSS later -- so, just concentrate on getting the appropriate form controls in here for this problem on this homework.
- an HTML comment containing the URL I can use to view your `bks-splash.html` from your nrs-projects account. (Note that, for full credit, this URL must successfully display your page when I paste it into a browser.)

Your resulting `bks-splash.html` file should now be ready to submit.