CS 325 - Homework 10

Deadline:

11:59 pm on **Friday, December 9, 2016**

How to submit:

Each time you wish to submit, within the directory `325hw10` on nrs-projects.humboldt.edu (and at the nrs-projects UNIX prompt, **NOT inside** `sqlplus`!) type:

```
~st10/325submit
```

...to submit your current files, using a homework number of `10`.

(**Make sure** that the files you intend to submit are listed as having been submitted!)

Purpose:

To think some more about the timestamps algorithm for concurrency control discussed in lecture, and to play lightly with a few of Oracle's date and string functions along with its simple report-related features.

Additional notes:

- SQL Reading Packet 8 and "regular" Reading Packet 10 are useful references for this homework.
- Now that we have covered the `order by` clause, you are expected to use it appropriately when an explicit row ordering is specified. Queries for problems asking for explicit row ordering will be incorrect if they do not include a reasonable `order by` clause.
- You are expected to follow the SQL style standards noted in previous homework handouts and mentioned in class.
- You are required to use the HSU Oracle `student` database for this homework.
- Feel free to add additional `prompt` commands to your SQL scripts as desired to enhance the readability of the resulting spooled output.
- An example `hw10-2-out.txt` has been posted along with this homework handout, to help you see if you are on the right track with your statements for Problem 2. If your `hw10-2-out.txt` matches this posted one, that doesn't guarantee that you wrote appropriate statements, but it is an encouraging sign.
  - (I added a few extra `prompt` commands near the beginning of this script to output some blank lines for slightly-prettier output.)

Problem 1

Place your name and then your answers for this part in a file named `hw10-1.txt` within your `325hw10` directory.

Consider the timestamps algorithm for concurrency control discussed in lecture. Please consider the following questions **independently** of one another.

1 part a

If the DBMS were to not permit one of these transactions to perform a desired action, what would it subsequently do to that transaction?

1 part b

Transaction T1, with timestamp 18, wishes to write a data item A, and the DBMS discovers that this data item was last read by a transaction with timestamp 8 (R-ts(A) = 8), and last written by a transaction with timestamp 7 (W-ts(A) = 7). Will the DBMS permit T1 to perform this action? Give R-ts(A) and W-ts(A) after this. (There should be three parts in your answer, then: will it be permitted, what is R-ts(A) afterwards, and what is W-ts(A) afterwards...)

1 part c

Transaction T2, with timestamp 19, wishes to write a data item B, and the DBMS discovers that this data item was last read by a transaction with timestamp 22 (R-ts(B) = 22), and last written by a transaction with timestamp 21 (W-ts(B) = 21). Will the DBMS permit T2 to perform this action? Give the R-ts(B) and the W-ts(B) after this.

1 part d

Transaction T3, with timestamp 13, wishes to read a data item C, and the DBMS discovered that this data item was last read by a transaction with timestamp 17 (R-ts(C) = 17), and last written by a transaction with timestamp 12 (W-ts(C) = 12). Will the DBMS permit T3 to perform this action? Give R-ts(C) and W-ts(C) after this.

1 part e

Transaction T4, with timestamp 24, wishes to read a data item D, and the DBMS discovers that this data item was last read by a transaction with timestamp 12 (R-ts(D) = 12), and last written by a transaction with timestamp 29 (W-ts(D) = 29). Will the DBMS permit T4 to perform this action? Give R-ts(D) and W-ts(D) after this.

1 part f

Transaction T5, with timestamp 18, wishes to write a data item E, and the DBMS discovers that this data item was last read by a transaction with timestamp 20 (R-ts(E) = 20), and last written by a transaction with timestamp 16 (W-ts(E) = 16). Will the DBMS permit T5 to perform this action? Give R-ts(E) and W-ts(E) after this.

Problem 2

Now, you are back to using Oracle and SQL again.

This problem again uses the tables created by the SQL script `hw4-create.sql` and populated by `hw4-pop.sql`. (And `hw4-pop.sql` still now includes appropriate `delete` commands so that it unpopulates the tables before trying to insert rows...!)

As a reminder, these create and populate a collection of tables that can be described in relation structure form as:

```
Movie_category(CATEGORY_CODE, category_name)

Client(CLIENT_NUM, client_lname, client_fname, client_phone, client_credit_rtg,
       client_fave_cat)
foreign key (client_fave_cat) references movie_category(category_code)

Movie(MOVIE_NUM, movie_title, movie_director_lname, movie_yr_released,
      movie_rating, category_code)
foreign key(category_code) references movie_category

Video(VID_ID, vid_format, vid_purchase_date, vid_rental_price, movie_num)
foreign key (movie_num) references movie

Rental(RENTAL_NUM, client_num, vid_id, date_out, date_due, date_returned)
foreign key (client_num) references client
foreign key(vid_id) references video
```

And, again, for your convenience as a reference, a handout of these relation structures is posted along with this homework handout.

(These tables should still exist in your database from Homework 4, so you should **not** need to re-run `hw4-create.sql` *unless* you have been experimenting with insertions or other table modifications, or unless any of these table names happen to conflict with your project tables...!)

Use `nano` (or `vi` or `emacs`) to create a file named `hw10-2.sql`:

```
nano hw10-2.sql
```

While within `nano` (or `vi` or `emacs`), type in the following:

- your name within a SQL comment
- `CS 325 Homework 10 - Problem 2` within a SQL comment
- the date this file was last modified within a SQL comment
- **don't start spooling yet**
  - (it will be specified below just where the `spool off` command should be placed, also...)

NOTE!!! READ THIS!!!

Now, within your file `hw10-2.sql`, add in SQL statements for the following, **PRECEDING** EACH ***EXCEPT* FOR PROBLEM 2-1** with a SQL*Plus `prompt` command noting what problem part it is for.

Problem 2-1

(This problem does **NOT** need to be preceded by a `prompt` command.)

Include SQL*Plus statements for each of the following within your script:

- explicitly clear any previously-set column headings
- create a **two line** top title and a **two line** bottom title (title contents of your choice)
- turn feedback off
- use `spool` to NOW start writing the results for the REST of this script's actions into a file `hw10-2-out.txt`

- put in a `prompt` command printing `Homework 10 - Problem 2`
- put in a `prompt` command printing your name

Problem 2-2

- change the pagesize to 20 lines
- write a `column` command to cause column `client_lname` to be formatted with heading `Client` and with width 13
- write a `column` command that will work for the date due displayed in the format given below, such that it will have a heading `Date Due` and a column width of 15. (You might want to write the query below, and then write this `column` command...)
- write a query that will display, for each rental that has not yet been returned:
  - the last name of the customer involved in that rental,
  - the date that rental is due, with the date displayed in the form `Month DD` (that is, for example, `September 6`, not including the year).
  - Display these in order of latest date due to earliest date due.

Problem 2-3

FUN FACT: `colsep` is the Oracle setting for what should come between columns --

```
show colsep
```

...shows that its default value is `" "` (a string containing one blank character), and you can set it using:

`set colsep` *desired_separator_string*

With this is mind...

- change the pagesize to 55 lines
- use `rpad` and `lpad` along with careful use of `colsep` and `column` commands...

  ...to write a query that projects two columns, the title of the movie on each video instance along with the rental price of that video, with the rows projected in order of the movie title and in secondary order of video rental price,

  ...such that there is a solid line of period characters (`.`) between the movie title and the video rental price

  ...and such that there is a `$` projected before the video rental price

  ...and such that the first column has the heading `MOVIE` and the second column has the heading `RENT FOR`
- then reset `colsep` back to its default value.
- HINT: concatenate the `'$'` to the `vid_rental_price` inside the call to the appropriate padding function
- HINT: use the `column` command to specify the width of each column

Problem 2-4

Write one, two, three, or four queries of your choice, using the tables from `hw4-create.sql`, that make use of your choice of **at least two** of the functions from the list:

`to_char` (using a DIFFERENT format string than you used in Problem 2-2)

`to_date   next_day   add_months   months_between`

AND **at least two** of the functions from the list:

`init_cap   lower   upper   ltrim   rtrim   length   substr`

...such that your chosen functions make a visible, noticeable difference in the query results.

(It is completely your choice whether you do additional formatting to these queries.)

Problem 2-5

Now:

- turn off your spooling
- either call `cleanup.sql` (available with this homework handout) or put in the SQL*Plus commands to at least:
  - clear columns
  - reset feedback to its default value,
  - reset pagesize to its default value,
  - turn off the top and bottom titles
  - (and clean up anything else you may have optionally set for Problem 2-4)

When you think the results of all of these problems look correct, this would also be a good time to look at the contents of `hw10-2-out.txt` -- at the nrs-projects prompt (the UNIX level, NOT in `sqlplus`!), type:

```
more hw10-2-out.txt
```

You should see that `hw10-2-out.txt` contains the problem results you just saw within `sqlplus`.

When you are satisfied with these, then `hw10-2.sql` and `hw10-2-out.txt` are completed.