CS 325 - Homework 8

Deadline:
11:59 pm on Thursday, December 6, 2013

How to submit:
When you are ready, within the directory 325hw8 on nrs-projects.humboldt.edu (and at the nrs-projects UNIX prompt, NOT inside sqlplus!) type:
~st10/325submit
...to submit this homework's files, using a homework number of 8. (Make sure that the files described below are listed as having been submitted!)

Purpose:
To think a little about database recovery, to think about some of the concurrency control mechanisms discussed in lecture, and to practice with some of SQL*Plus commands related to creating ASCII reports

Additional notes:
• You are required to use the HSU Oracle student database 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.
• An example hw8-results.txt has been posted along with this homework handout, to help you see if you are on the right track with your SQL statements. If your hw8-results.txt matches this posted one, that doesn't guarantee that you wrote appropriate queries, but it is an encouraging sign...
  – Be careful, though -- in some cases, there is more than one way to write a query that gives the same results, but for some of this homework's problems you are required to write a query that gets those results using certain query approaches (and perhaps deliberately not using other query approaches). (And, of course, your queries always need to meet course SQL style guidelines.) You need to meet those specifications, meet those style guidelines, and get the desired results for such problems.

The Problems

Problem 1
Place your answers for this part in a file named hw8-1.txt within your 325hw8 directory.

Problem 1-1
You have a transaction log, in which all transaction operations since the last complete database backup are recorded in chronological order. A failure occurs at time X, and recovery is begun by starting with the last
complete database state. In this system, recovery is done using rollforward, as discussed in lecture.

**1-1 part a**
In looking at the transaction log during recovery, it is discovered that transaction T1 had been rolled
back/aborted before time X. Will its logged actions be re-done as part of the recovery process?

**1-1 part b**
In looking at the transaction log during recovery, it is discovered that transaction T2 had been committed
before time X. Will its logged actions be re-done as part of the recovery process?

**1-1 part c**
In looking at the transaction log during recovery, it is discovered that transaction T3 had not yet been
committed before time X. Will its logged actions be re-done as part of the recovery process?

**For Problems 1-2 through 1-5...**
Assume that A, B, C, D, and E are data items and that T1, T2, T3, T4, T5, and T6 are transactions.

**Problem 1-2**
Assume that your DBMS is using binary locks for concurrency control. Currently, T1 has a binary lock on
data item A, and T2 has a binary lock on data item C. All other data items are unlocked.

**1-2 part a**
T3 requests a binary lock on B. Will T3 obtain the lock at this point, or will T3 have to wait?

**1-2 part b**
T4 requests a binary lock on C. Will T4 obtain the lock at this point, or will T4 have to wait?

**Problem 1-3**
Assume that your DBMS is using shared/exclusive locks for concurrency control. Currently T1 has an
exclusive/write lock on data item A, and T2 has a shared/read lock on data item C. All other data items are
currently unlocked.

**1-3 part a**
Consider the operations Read and Write. Which of these can T1 do to A, given its lock (read, write, neither
read nor write, or both read and write)?

**1-3 part b**
Consider the operations Read and Write. Which of these can T2 do to C, given its lock (read, write, neither
read nor write, or both read and write)?
1-3 part c
T3 requests an exclusive lock on D. Will T3 obtain the lock at this point, or will T3 have to wait?

1-3 part d
T4 requests a shared lock on C. Will T4 obtain the lock at this point, or will T4 have to wait?

1-3 part e
T5 requests an exclusive lock on C. Will T5 obtain the lock at this point, or will T5 have to wait?

1-3 part f
T6 requests a shared lock on B. Will T6 obtain the lock at this point, or will T6 have to wait?

Problem 1-4
Transaction T1 has an exclusive lock on data item A, and has requested, and is waiting for, a shared lock on data item B. But, T2 has an exclusive lock on data item B, and has requested, and is waiting for, an exclusive lock on data item A. What is this situation an illustration of?

Problem 1-5
Consider the timestamps algorithm for concurrency control discussed in lecture. Please consider the following questions independently of one another.

1-5 part a
If the DBMS were to not permit one of these transactions to perform a desired action, what would it subsequently need to do to that transaction?

1-5 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...)

1-5 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.

1-5 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.

1-5 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.

1-5 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

This homework again uses the tables created by the SQL script \texttt{hw3-setup.sql}. As a reminder, it created and populated 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)

\hspace{1cm} 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)

\hspace{1cm} foreign key(category\_code) references movie\_category

Video(VID\_ID, vid\_format, vid\_purchase\_date, vid\_rental\_price, movie\_num)

\hspace{1cm} foreign key (movie\_num) references movie

Rental(RENTAL\_NUM, client\_num, vid\_id, date\_out, date\_due, date\_returned)

\hspace{1cm} foreign key (client\_num) references client

\hspace{1cm} foreign key(vid\_id) references video

Use \texttt{nano (or vi or emacs)} to create a file named \texttt{hw8-2.sql}:

\texttt{nano hw8-2.sql}

While within \texttt{nano (or vi or emacs)}, type in the following:

\begin{itemize}
  \item your name within a SQL comment
  \item 325 Homework 8 - Problem 2 as a SQL comment
  \item the date this file was last modified as a SQL comment
  \item don't start spooling yet
  \item (it will be specified below just where the spool off command should be placed, also...)
\end{itemize}

Now add in SQL statements for the following, \textbf{preceding each EXCEPT FOR PROBLEMS 2-1 and 2-2 with a prompt command noting what question it is for.} (I don't mind if you ALSO put a comment, but
the prompt command improves your spooled output's readability.)

**Problem 2-1**

(This problem does not need to be preceded by a prompt command.)

Because this script includes update and delete statements, this script should start with a "fresh" set of tables each time it runs.

- Make a copy of hw3-setup.sql in your 315hw8 directory
- place a call executing hw3-setup.sql

**Problem 2-2**

(This problem also 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, breaks, and computes
- create a two line top title and a two line bottom title (title contents of your choice)
- make the pagesize 35 lines and the linesize 75 characters.
- turn feedback off
- now begin spooling to hw8-results.txt

**Problem 2-3**

(NOW start preceding each problem with a prompt command.)

Consider the rental_history view from Homework 7, Problem 2-5.

Write column commands for each of the following:

- give column client_name the heading Client (camel-case, uppercase for the first letter and lowercase for the rest), and format it so that it is:
  - narrower than its default width,
  - but wide enough for all of the last-name-first name combinations currently there,
  - but narrow enough that all of the columns "fit" on 1 line in the subsequent query results involving this column

- give column movie_title the heading Movie Title (camel-case, including the blank between the 2 words), and format it so that it, too, is:
  - narrower than its default width,
  - but wide enough for all of the movie titles currently there,
  - but narrow enough that all of the columns "fit" on 1 line in the subsequent query results involving this column

- give column vid_format a 2-line heading, with Video on the first line and Format on the second (and each in camel-case as shown), and format it so that its entire heading shows
*  give column vid_rental_price a 2-line heading, with Rental on the first line and Price on the second (and each in camel-case as shown), and format it so that:
  *  it contains a $
  *  it always displays to 2 fractional places (to 2 decimal places)
  *  its entire heading shows

Follow these column commands with a query showing all of the current contents of the view rental_history, displaying the rows in order of most expensive video rental price to least expensive video rental price, with a secondary ordering by client_name, and with a third ordering by movie_title.

**Problem 2-4**

Consider the category_stats view from Homework 7, Problem 2-6.

First, change the pagesize to 20 lines.

Write column commands for each of the following:

Write column commands for each of the following:

*  give category_stats' first column the heading Category (in camel-case as shown)
*  give category_stats' second column the heading # Videos (camel-case, including the blank between the 2 parts), and format it so that it is wide enough to show the whole heading
*  give category_stats' third column a 2-line heading, with Average on the first line and Price on the second (and each in camel-case as shown), and format it so that:
  *  it contains a $
  *  it always displays to 2 fractional places (to 2 decimal places)
  *  its entire heading shows

Follow these column commands with a query showing all of the current contents of the view category_stats, displaying the rows in order of most number of videos to least number of videos, with a secondary ordering by highest average rental price to lowest average rental price.

**Problem 2-5**

Do the following:

*  change the pagesize to 60 lines
*  write a break command to suppress repeated movie titles, putting one blank line between each set of such rows (note that a break command does not have to include a table name -- and indeed it would be better to avoid using a table name here, since the query below involves a join)
*  write a column command using like to cause column client_lname to be formatted like the column client_name (WITHOUT specifically repeating the heading and format clauses of client_name's column command)
*  write a query to project the titles of movies that have been rented and the last names of the clients who rented them, displaying the rows in order of movie title and in secondary order by client last name
Problem 2-6

Do the following:

* change the pagesize to 45 lines
* write a break command to suppress repeated category names and movie ratings, putting one blank line between each set of such category names (but NOT between each set of such movie ratings) (this is another break where you should avoid including a table name)
* write a column command to set the category_name's column to have a display heading of Category (in camel-case as shown)
* write a column command to set the movie_rating's column to have a display heading of Rating (in camel-case as shown) with a format wide enough that the entire heading shows
* write a query to project the movie category name, the movie rating, and the movie title for each movie, displaying the rows in order of category name, in secondary order by movie_rating, and in third order by movie_title

Problem 2-7

Consider Problem 2-6. Write a compute command that will cause how many movie_titles are within each set of consecutive category names to be displayed, and then use / to redo the previous query, which should now include these counts.

Problem 2-8

First, clear computes. Then:

* write a break command to suppress repeated category names and movie titles, putting one blank line between each set of such category names (but NOT between each set of such movie titles) (this is another break where you should avoid including a table name)
* write a compute command that will cause the average video rental price for each set of such category names to be displayed
* write a column command that will set the vid_rental_price's column to have a display heading of Cost (in camel-case as shown) with a format that always displays with 2 fractional places, such that prices below $1.00 appear with a 0 before the decimal point (e.g., 0.99), and wide enough that the entire heading and the prices show.
* write a query to project the category name, the movie title, and the video rental price for each video rental, displaying the rows in order of category name, in secondary order by movie title, and in third order by video rental price.

Problem 2-9

Change the pagesize to 70.

* write a column command that first specifies a format for the movie_title column giving it a width of 11. Then write a query to project just the movie titles, displaying them in alphabetical order.
* now write a column command that specifies a format for the movie_title column giving it a width of 11, but now also specifies that it should be word-wrapped. Then use / to redo the previous query,
which should now display using this word-wrapping.

* change the pagesize to 30
* finally, write a column command that specifies a format for the movie_title column giving it a width of 11, but now also specifies that it should be truncated. Then use / to redo the previous query, which should now display using this truncation.

Problem 2-10

* clear breaks
* write a column command that specifies a format for the movie_title column that is wide enough for all of the movies currently in that column.
* 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...)
* Then, 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 name of the movie involved in that rental, and
  * 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 2-11

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, breaks, and computes,
  * reset feedback to its default value,
  * reset pagesize and linesize to their default values,
  * turn off the top and bottom titles

When you think the results of all of these queries look correct, this would also be a good time to look at the contents of hw8-results.txt -- at the nrs-labs prompt (the UNIX level, NOT in sqlplus!), type:

more hw8-results.txt

You should see that hw8-results.txt contains the query results you just saw within sqlplus.

When you are satisfied with these, then hw8-2.sql and hw8-results.txt are ready to submit.