CIS 315 - Homework 7

Deadline:

NOTE - UNUSUAL DEADLINE! (because I waited, hoping we'd get word about sqlldr...)

11:59 pm on FRIDAY, November 19th

How to submit:

When you are ready, within the directory `315hw7` on `nrs-labs.humboldt.edu` (and at the nrs-labs UNIX prompt, NOT inside `sqlplus`!) type:

```
~st10/315submit
```

...to submit these `.sql` and `.txt` files, using a homework number of `7`. (**Make sure** that you see the names of all of the files you wished to submit!)

Purpose:

To think a little about integrities and database recovery, to practice with SQL views, and to practice with a few 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 `hw7-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 `hw7-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 `hw7-1.txt` within your `315hw7` directory.

Problem 1-1

We have discussed entity integrity, domain integrity, referential integrity, and transaction integrity, and how a DBMS may support (or enforce) them.

Give the most appropriate of these four types of integrity in answer to each of the following:

1-1 part a

This is being supported (to at least some degree) in the Oracle DBMS by refusal to allow the insertion of an attribute value that is not the given type for that attribute (based on the type declaration for that attribute).

(that is, this kind of integrity is being enforced when the Oracle DBMS will not permit the insertion of a row that includes a date value for a column declared to be of type integer)

1-1 part b

This is being supported when a DBMS does not permit the insertion of a row with the same primary key as an already-existing row in that table.

1-1 part c

This is being supported when a DBMS does not permit the insertion of a row with a foreign key value for which there is not a corresponding primary key value in the parent table.

1-1 part d

Oracle SQL supports this with its COMMIT and ROLLBACK statements.

1-1 part e

If a DBMS permitted you to insert duplicate rows within a table, it would not be supporting this.

1-1 part f

One way in which Oracle DBMS supports this is with its "on delete restrict" default for foreign keys --- that is, in Oracle, if you attempt to delete a parent row with a primary key matching children rows with that value for their foreign key, you are not permitted to do so.

1-1 part g

When a DBMS has automatic backup and recovery capabilities, it could be said to be providing support

for this (in terms of the desired property of durability).

1-1 part h

Oracle is providing additional support for this with the check clauses that one may use in a create table statement to restrict the allowed values in a particular column, which it then uses to only permit insertions of rows which have values for that column that satisfy the check clause.

Problem 1-2

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.

1-2 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?

1-2 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?

1-2 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?

Problem 2

This homework again uses the tables created by the SQL script `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)
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

Use `nano` (or `vi` or `emacs`) to create a file named `hw7-2.sql`:

```
nano hw7-2.sql
```

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

- your name within a SQL comment
- `315 Homework 7 - Problem 2` as a SQL comment
- the date this file was last modified as a SQL comment
- **don't start spooling yet**
- (it will be specified below just where the spool off command should be placed, also...)

Now add in SQL statements for the following, **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 `315hw7` directory
- place a call executing `hw3-setup.sql`. (That is, place the command you would type within `sqlplus` to run `hw3-setup.sql` within your script `hw7-2.sql` BEFORE it starts spooling to `hw7-results.txt`)
  - (why? because we really don't need to see all of the row-inserted feedbacks in our result file...)

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:

* make the pagesize 35 lines and the linesize 100 characters.
* turn feedback off
* **now** begin spooling to `hw7-results.txt`

Problem 2-3

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

Drop and create a view called `mini_action` which contains the columns `movie_num`, `movie_title`, and `movie_rating` from the `movie` table for rows with `category_code` of `200`. Follow that with a query showing all of the contents of this view, displaying the rows in order of

`movie_title`.

Problem 2-4

Drop and create a view called `movie_list` of the `movie` and `movie_category` tables, containing only the category name, movie rating, and movie title for each movie.

Follow that with a query showing all of the contents of this view, displaying the rows in order of movie rating, with a secondary ordering by movie title.

Problem 2-5

Drop and create a view called `rental_history` which gives a view of which clients have rented which videos by including rows with the following 4 columns:

* in its first column, it has the last name of the client followed by the first name, with a comma and a blank separating the last name and the first name of each (e.g.:

```
Tuttle, Sharon
```

). Name this column `client_name` (do not use double-quotes in specifying this column name, or any of the view column names given in this problem)

* in its second column, it should have the movie title rented in that rental; make sure that, one way or another, the name of this column is `movie_title`

* in its third column, it should have the format of the video rented; make sure that, one way or another, the name of this column is `vid_format`

* in its fourth column, it should have the `vid_rental_price`; make sure that, one way or another, the name of this column is `vid_rental_price`

Follow that with a query showing all of the contents of this view, 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 2-6

Consider what you get when you perform an equi-join of the `movie`, `video`, and `movie_category` tables --- now you have the corresponding movie details for each video's movie. Drop and create a view called `category_stats` which groups the videos by movie category name and contains only the category name, the number of videos in that category, and the average rental price of videos in that category. (Remember to rename the column names corresponding to function calls; choose appropriate column names.)

Follow that with a query showing all of the contents of this view, 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 2-7

2-7 part a

Write an `update` command to decrease the rental prices of all videos with format Blu-Ray by 0.99. Then display the current contents of the view `category_stats` again, using the same row-ordering as you did at the end of Problem 2-6. (Note that / will not work for redoing this query -- it causes the last SQL command to be redone, which in this case is the `update` command, which we do **NOT** want redone...!)

2-7 part b

Write a query, using the view `rental_history` only, performing a "true" relational projection of just the names of those clients who have rented 'Gone with the Wind', displaying the rows in order of `client_name`.

2-7 part c

Write a single `delete` command that will delete all rows from `rental` that involve the client with client number '5555'. Then repeat your query from part b.

2-7 part d

Write a query, using the view `rental_history` only, to project each client's name and the number of rentals they have made, displaying the rows in order from highest to lowest number of rentals, with a secondary ordering by client name.

Problem 2-8

Now:

* turn off your spooling
* reset feedback to its default value
* reset pagesize and linesize to their default values

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 `hw7-results.txt` -- at the nrs-labs prompt (the UNIX level, NOT in `sqlplus`!), type:

```
more hw7-results.txt
```

You should see that `hw7-results.txt` contains the query results you just saw within `sqlplus`.

When you are satisfied with these, then `hw7-2.sql` and `hw7-results.txt` are ready to submit.