CS 325 - Homework 2

Deadline:

Problem 1 -- answering reading questions on Moodle for Reading Packet 3 -- had to be completed by 10:45 am on Tuesday, September 13.

Problem 2 -- answering reading questions on Moodle for Reading Packet 4 -- needs to be completed by 10:45 am on Tuesday, September 20.

The remaining problems are due by **11:59 pm** on **Friday, September 23, 2016**.

How to submit:

For Problem 3 onward:

Each time you wish to submit, within the directory `325hw2` 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 `2`.

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

Purpose:

To see if you are gleaning some important concepts from required class reading, to lightly try writing `grant` and `revoke` statements, to practice with additional domain constraints, to think about how DBMS's help support various levels of database integrity, to practice writing `insert` statements that demonstrate such DBMS integrity support including supporting domain constraints, and to practice thinking about and writing relational operations, for this homework "by hand" (NOT YET using SQL!!).

Additional notes:

- You are required to use the HSU Oracle `student` database for this homework.
- SQL Reading Packet 1 and "regular" Reading Packet 3, on the course Moodle site, are useful references 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.

Problem 1

Had to correctly answer the Reading Questions for Reading Packet 3, on the course Moodle site, by 10:45 am on Tuesday, September 13.

Problem 2

Have to correctly answer the Reading Questions for Reading Packet 4, on the course Moodle site, by 10:45 am on Tuesday, September 20.

Setup for Problems 3 onward

Use `ssh` to connect to `nrs-projects.humboldt.edu`, and create a directory named `325hw2` on nrs-projects:

```
mkdir 325hw2
```

...and change this directory's permissions so that only you can read it:

```
chmod 700 325hw2
```

...and change your current directory to that directory (go to that new directory) to do this homework:

```
cd 325hw2
```

Put all of your files for this homework in this directory. (And it is from this directory that you should type `~st10/325submit` to submit your files when you are done.)

Problem 3

I don't really want you to grant access to any of your tables to someone else at this point -- but I do want you to practice writing a few `grant` and `revoke` commands. Fortunately, I know of a couple of users' usernames you can practice with that should be quite safe.

Use `nano` (or `vi` or `emacs`) to create a file named `hw2-3.sql`:

```
nano hw2-3.sql
```

While within `nano` (or whatever), type in the following within one or more SQL comments:

- your name
- `CS 325 Homework 2-3`
- the date this file was last modified

Then:

- use `spool` to start writing the results for this script's actions into a file `hw2-3-out.txt`
- include a `spool off` command, at the BOTTOM/END of this file. Type your answers to the parts below BEFORE this spool off command!

NOTE!!! READ THIS!!!

Now, within your file `hw2-3.sql`, add in statements for the following, **PRECEDING** EACH with a SQL*Plus `prompt` command noting what problem part it is for.

Problem 3 part a

You determine that you would like users `st10` and `dt5` to be able to see the contents of your table `video`, but that is ALL they should be able to do with the `video` table -- they shouldn't be able to change it in any way.

Write an appropriate `grant` statement that would have this result.

Problem 3 part b

You also decide that you would like users `st10` and `dt5` to be able to change column values of rows already in your `client` table, and to be able to see its contents, but that is ALL they should be able to do.

Write an appropriate `grant` statement that would have this result.

Problem 3 part c

And, you decide that you would like users `st10` and `dt5` to be able to do anything they would like with the contents of your `rental` table (including adding rows, getting rid of rows, changing their contents, and seeing its contents).

Write an appropriate `grant` statement that would have this result.

Problem 3 part d

...And now you have a change of heart, and want to remove all access you have granted to `st10` and `dt5` from your tables in Problem 3 parts b, c, and d.

Write appropriate `revoke` statements that would have this result.

If you haven't already, save your `hw2-3.sql` file and go into `sqlplus` and see if `start hw2-3.sql` works. Do the SQL statement results look correct? (Mind you, "look correct" in this case is simply feedback letting you know that the `grant` and `revoke` statements succeeded...!)

When you think the results look correct, this would also be a good time to look at the contents of `hw2-3-out.txt` -- at the nrs-projects prompt (the UNIX level, NOT in `sqlplus`!), type:

```
more hw2-3-out.txt
```

You should see that `hw2-3-out.txt` contains the results you just saw within `sqlplus`.

When you are satisfied with these, then `hw2-3.sql` and `hw2-3-out.txt` are completed.

Problem 4

Use `nano` (or `vi` or `emacs`) to create a file named `hw2-4.sql`:

```
nano hw2-4.sql
```

While within `nano` (or whatever), type in the following within one or more SQL comments:

- your name
- `CS 325 Homework 2-4`
- the date this file was last modified

Then:

- use `spool` to start writing the results for this script's actions into a file `hw2-4-out.txt`
- include a `spool off` command, at the BOTTOM/END of this file. Type your answers to the parts below BEFORE this spool off command!

NOTE!!! READ THIS!!!

Now, within your file `hw2-4.sql`, add in statements for the following, **PRECEDING** EACH with a SQL*Plus `prompt` command noting what problem part it is for.

Problem 4 part a

Consider your `client`, `video`, and `rental` tables from Homework 1. You are going to **re-create** these with some additional domain constraints.

(Note that they should have the same columns and the same primary and foreign keys as before -- you are just adding additional domain constraints to the declarations for some of their columns.)

Put in appropriate `drop table` and `create table` statements for `client`, `video`, and `rental` such that:

- in the `client` table, every row **must** have a last name and a phone number (neither of these can be empty)
- in the `video` table,
  - the video format must be either DVD, HD-DVD, or BluRay (it must be one of these; the video format cannot be empty)
  - the video purchase date should have a default value of the current date (using `sysdate`) if no video purchase date is explicitly given
  - the video rental price must be greater than 0, and cannot be empty
  - the video length must be greater than 0 (but it is OK if it is empty)
- re-create the `rental` table as it was in Homework 1

Then, copy in your `insert` statements from Homework 1 for all three of these tables; all *should* still work (although modify your additional client and video row if you happened to include values that now violate any of new domain constraints)

Problem 4 part b

Write a `prompt` that prints `SHOULD FAIL; VIOLATES ENTITY INTEGRITY` to the screen.

Follow this with an `insert` statement for either `client`, `video`, or `rental` that fails *specifically*

because the Oracle DBMS is enforcing **entity** integrity.

(Remember: examples of a DBMS enforcing entity integrity include it not allowing null primary keys, and not allowing multiple rows in a table to have the same value for the primary key.)

Problem 4 part c

Write a `prompt` that prints `SHOULD FAIL; VIOLATES REFERENTIAL INTEGRITY` to the screen.

Follow this with an `insert`, `update`, OR `delete` statement for either `client`, `video`, or `rental` that fails *specifically* because the Oracle DBMS is enforcing **referential** integrity.

Problem 4 part d

Write a `prompt` that prints `SHOULD FAIL; VIOLATES DOMAIN INTEGRITY` to the screen.

Follow this with an `insert` statement for either `client`, `video`, or `rental` that fails *specifically* because the Oracle DBMS is enforcing **domain** integrity in particular (that couldn't also be reasonably described as enforcing entity or referential integrity also) .

Problem 4 part e

Write a `prompt` that prints `SHOULD FAIL; client needs phone` to the screen.

Follow this with an `insert` statement for `client`, using the version of `insert` where you give the names of just the columns being filled, that tries to create a client with **just** a reasonable client id and last name.

Problem 4 part f

Write a `prompt` that prints `SHOULD FAIL; video needs legal format` to the screen.

Follow this with an `insert` statement for `video` that tries to create a video with reasonable values for all of its columns EXCEPT it tries to give it a video format of `'MOO'`.

Problem 4 part g

Write an `insert` statement for `video`, using the version of `insert` where you give the names of just the columns being filled, that creates a video with reasonable values for all of its columns EXCEPT it does not explicitly specify a value for the video purchase date column.

Problem 4 part h

Write a `prompt` that prints `SHOULD FAIL; video can't have negative price, length` to the screen.

Follow this with two `insert` statements for `video` that try to create two videos with reasonable values for all of their columns EXCEPT one tries to specify a negative video rental price and the other tries to specify a negative video length.

Problem 4 part i

Now write three `select` statements displaying the contents of the tables `client`, `video`, and `rental` at this point.

Are there any other `insert`, `update`, or `delete` statements, that should succeed or should not, that you would like to try? If so, write a `prompt` that prints `MORE EXPERIMENTS:` to the screen, followed by your experiments.

When you think the results look correct, this would also be a good time to look at the contents of `hw2-4-out.txt` -- at the nrs-projects prompt (the UNIX level, NOT in `sqlplus`!), type:

```
more hw2-4-out.txt
```

You should see that `hw2-4-out.txt` contains the results you just saw within `sqlplus`.

When you are satisfied with these, then `hw2-4.sql` and `hw2-4-out.txt` are completed.

Problem 5

NOTE: THIS PROBLEM DOES NOT USE ORACLE OR SQL AT ALL!!

Use `nano` (or `vi` or `emacs`) to create a file named `hw2-5-by-hand.txt`:

```
nano hw2-5-by-hand.txt
```

While within `nano` (or whatever), type in your name, and then your answers, in plain text, for the following (preceding each answer with the number of the question being answered). Note that, if you are asked to give a relation in tabular form as your answer, you should do so by putting the attribute names on one line, a row of dashes , and then the attribute values neatly lined up underneath.

Consider the tables `Client`, `Video`, and `Rental` -- for this problem, assume that their current contents are:

the Client relation:

Cli_id	Cli_lname	Cli_fname	Cli_phone
000A	Alpha	Ann	000-0001
111B	Beta	Bob	111-1112
222B	Beta	Ann	222-2223
333C	Carlos	David	333-3334
444D	Delta	Edie	111-1112

the Video relation:

Vid_id	Vid_format	Vid_purchase_date	Vid_rental_price	Vid_length
00000D	DVD	11-JAN-2013	1.99	73
11111H	HD-DVD	22-FEB-2014	4.99	91
22222B	BluRay	03-MAR-2012	1.99	105
33333H	HD-DVD	22-FEB-2014	3.99	69
44444B	BluRay	04-APR-2010	0.99	91

the Rental relation:

Cli_id	Vid_id
111B	11111H
222B	00000D
222B	22222B
333C	22222B
333C	00000D
333C	11111H
000A	44444B

Problem 5 part a

Mentally perform a relational selection (by hand, NOT using SQL!) of the rows of `Video` for which the `Vid_format` is `'HD-DVD'`, typing in tabular form the relation that results.

Problem 5 part b

Mentally perform a relational projection (by hand, NOT using SQL!) of the `Vid_purchase_date` and `Vid_format` attributes of `Video`, typing in tabular form the relation that results.

Problem 5 part c

Mentally CONSIDER the Cartesian product of the `client` and `video` relations (don't DO it, just CONSIDER it). **How many rows** would be in the resulting relation?

Problem 5 part d

Mentally perform a relational natural join (by hand, NOT using SQL!) of the relations `Video` and `Rental` on the attribute `vid_id`, (using the join condition `Video.vid_id = Rental.vid_id`), typing in tabular form the relation that results. Note that you only need to give the final resulting relation, **not** the intermediate steps along the way.

IMPORTANT for Problem 5 parts e through h

Consider the relational operators:

- selection
- projection
- Cartesian product
- equi-join
- natural join

For each of Problem 5 parts e through h, give the **name** of the most appropriate (**single**) **relational operator** that could be used to result in the desired relation. (I am **only** asking for the appropriate operation's **NAME**, here.)

Problem 5 part e

Which (**single**) relational operator could be used to list **just** the client last names and client phone numbers from `client`?

Problem 5 part f

Which (**single**) relational operator could be used to result in a **single** relation containing **only** the attributes `vid_id`, `vid_format`, `vid_purchase_date`, `vid_rental_price`, `vid_length`, `cli_id`?

Problem 5 part g

Which (**single**) relational operator could be used to result in a relation containing the attributes `video.vid_id`, `vid_format`, `vid_purchase_date`, `vid_rental_price`, `vid_length`, `rental.vid_id`, `cli_id`? (Careful -- Problem 5 part f and Problem 5 part g have slightly different answers!)

Problem 5 part h

Which (**single**) relational operator could be used to result in a relation containing all of the `Video` attributes, but only for videos whose rental price is more than $2 (that is, the resulting relation has the attributes `vid_id`, `vid_format`, `vid_purchase_date`, `vid_rental_price`, and `vid_length`, but it only contains that information for videos whose `vid_rental_price` is more than $2).

`hw2-5-by-hand.txt` is now completed.