CIS 315 - Homework 2

Deadline:

1:00 pm (beginning of lab) on Wednesday, September 22nd

How to submit:

When you are ready, within the directory `315hw2` 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 `2`. (You should see **5** files being submitted!)

Purpose:

To practice writing relations in relation-structure form, practice specifying foreign keys, and practice thinking about and writing relational operations, "by hand" and using SQL.

Additional notes:

- You are required to use the HSU Oracle `student` database for this homework.

Problem 0:

Use `ssh` to connect to `nrs-labs.humboldt.edu`, and create a directory named `315hw2` on nrs-labs:

```
mkdir 315hw2
```

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

```
chmod 700 315hw2
```

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

```
cd 315hw2
```

Put all of your files for this homework in this directory. (And it is from this directory that you should type `~st10/315submit` to submit your files when you are done.)

Use `nano` (or `vi` or `emacs`) to create a file named `hw2-1.txt` within directory `315hw2`:

```
nano hw2-1.txt
```

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

- your name
- `315 Homework 2-1`

- the date this file was last modified

Problem 1:

Consider the following relations, written in **tabular form** -- note that we are now adding a third table, a **rental** table. Also, note that, in this bizarre scenario, a client is only allowed to rent a particular video one time, ever. (That's an example of a business rule, by the way!)

the Client relation:

Cli_num	Cli_lname	Cli_fname	Cli_phone
0000	Alpha	Ann	000-0001
1111	Beta	Bob	111-1112
2222	Beta	Ann	222-2223
3333	Carlos	David	333-3334
4444	Delta	Edie	111-1112

the Video relation:

Vid_id	Vid_format	Vid_purchase_date	Vid_rental_price	Vid_length
000000	Beta	11-JAN-1998	1.99	73
111111	DVD	22-FEB-1999	4.99	91
222222	VHS	03-MAR-1997	1.99	105
333333	DVD	22-FEB-1999	3.99	69
444444	VHS	04-APR-1994	0.99	91

the Rental relation:

Cli_num	Vid_id
1111	111111
2222	000000
2222	222222
3333	222222
3333	000000
3333	111111
0000	444444

In `hw2-1.txt`, write these tables in **relation structure** form, writing the primary key attributes in all-uppercase. (Part of your task here is to determine what would be a **reasonable** primary key for each of these tables.)

(Remember: the primary key must **uniquely** identify a row, and it can consist of one or more attributes.)

(Example relation structure:

```
Parts(PART_NUM, part_name, quant_on_hand, price, level_code,
      last_inspected)
```

...for table `Parts`, with attributes `part_num`, `part_name`, `quant_on_hand`, `price`, `level_code`, and `last_inspected`, with primary key consisting of attribute `part_num`, because `part_num` uniquely determines a row (every part has a unique part number).)

Save this file when you are done; it is now ready to submit. (MAKE SURE ITS NAME ENDS WITH `.txt`)

Problem 2:

Use `nano` (or `vi` or `emacs`) to create a file named `hw2-2.sql`:

```
nano hw2-2.sql
```

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

- your name within a SQL comment
- `315 Homework 2-2` as a SQL comment
- the date this file was last modified as a SQL comment

Now, within your file `hw2-2.sql`:

- copy over your drop-table, create-table, and insert statements for `client` and `video` from Homework 1 (because we need to modify them); make sure you include the additional `client` row and `video` row you inserted as part of Homework 1.
- change their drop-table statements to now end with `cascade constraints` (and don't forget the semicolons after that!)
- add drop- and create-table statements for table `rental`, also, and add insert statements to populate it with the rows shown on p. 2
  - make sure that `rental` is created with appropriate primary and foreign keys
- be careful about the order of your table creations; when foreign keys are involved, you'll see that you have to create the "parent" table before creating the "child" table when a foreign key is involved (the "parent" has to exist to be referenced by the "child"'s foreign key). So make sure that you create the `client` and `video` tables before you create the `rental` table.
  - (Oh yes -- and make sure you populate the `client` and `video` tables before you populate the `rental` table, too, for similar reasons.)
- insert one additional row into the `rental` table which has the additional client you added in Homework renting the additional video you added in Homework 1

This would be a good time (if you haven't already) to save your `hw2-2.sql` file, go into `sqlplus` and see if `start hw2-2.sql` works -- are the 3 tables successfully dropped and created?

`hw2-2.sql` is now ready to submit.

Problem 3:

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

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

```
nano hw2-3-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.

Problem 3-1

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

Problem 3-2

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 3-3

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 3-4

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.

Problem 3-5

For each of the following, 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 3-5, part a

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

Problem 3-5, part b

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

Problem 3-5, part c

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, client_num? (Careful --- (b) and (c) have slightly different answers!)

Problem 3-5, part d

Which 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).

`hw2-3-by-hand.txt` is now ready to submit.

Problem 4:

YOU ARE USING ORACLE AGAIN NOW...

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:

- your name within a SQL comment
- `315 Homework 2-4` as a SQL comment
- the date this file was last modified as a SQL comment
- use `spool` to start writing the results for this script's actions into a file `hw2-results.txt`
- include a `spool off` command, at the BOTTOM/END of this file. Type your answers to the problems below BEFORE this spool off command!

Now, within your file `hw2-4.sql`, add in statements for the following:

Problem 4-1

Write a SQL statement to perform a "pure" relational projection of the video format and video rental price columns only (and in that order) from the `video` table.

Problem 4-2

Write a SQL statement to perform a relational selection of rows of the `video` table for videos with a length of more than 75 minutes.

Problem 4-3

Write a SQL statement to perform a relational selection of rows of the `rental` table for rentals involving the video with ID of `'000000'`.

Problem 4-4

Write a SQL statement to perform a Cartesian product of the `rental` and `video` tables.

Problem 4-5

Write a SQL statement to perform an equi-join of the `rental` and `video` tables.

Problem 4-6

Write a SQL statement to perform a natural join of the `rental` and `client` tables.

Problem 4-7

Write a SQL statement that projects just the client's last name and the video ID's from the equi-join of the `rental` and client tables.

If you haven't already, save your `hw2-4.sql` file and go into `sqlplus` and see if `start hw2-4.sql` works. Do the SQL statement results look correct?

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

```
more hw2-results.txt
```

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

When you are satisfied with these, then `hw2-4.sql` and `hw2-results.txt` are ready to submit.