CS 325 - Homework 3

Deadline:

Problem 1 -- answering reading questions on Moodle for Reading Packet 5 -- needs to be completed by 10:45 am on Tuesday, September 27.

The remaining problems are due by **11:59 pm** on **Friday, September 30, 2016**.

How to submit:

For Problem 2 onward:

Each time you wish to submit, within the directory `325hw3` 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 `3`.

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

Purpose:

To practice thinking about and writing relational operations using SQL, and to practice with database modeling (creating an ERD meeting the required notation for this course).

Additional notes:

- You are required to use the HSU Oracle `student` database for this homework.
- SQL Reading Packet 2 and "regular" Reading Packet 4, 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.
- NOTE the following course style standards for SQL `select` statements:
  - In a SQL script, put a blank line BEFORE and AFTER each `select` statement, for better readability.
  - A `select` statement's `from` clause should ALWAYS start on a new line.
  - If a `select` statement has a `where` clause, it should always start on a new line.
  - If a clause is longer than one line, INDENT the continuation on the next by at least three spaces (so it is clear which clause it "belongs" to).

Problem 1

Have to correctly answer the Reading Questions for Reading Packet 5, on the course Moodle site, by 10:45 am on Tuesday, September 27.

Setup for Problems 2 onward

Use `ssh` to connect to `nrs-projects.humboldt.edu`, and create a directory named `325hw3` on nrs-projects:

```
mkdir 325hw3
```

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

```
chmod 700 325hw3
```

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

```
cd 325hw3
```

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 2

YOU ARE USING ORACLE and SQL FOR THIS PROBLEM.

Use `nano` (or `vi` or `emacs`) to create a file named `hw3-2.sql`:

```
nano hw3-2.sql
```

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

- your name within a SQL comment
- `CS 325 Homework 3-2` within a SQL comment
- the date this file was last modified within a SQL comment
- use `spool` to start writing the results for this script's actions into a file `hw3-out.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!

NOTE!!! READ THIS!!!

Now, within your file `hw3-2.sql`, add in statements for the following, **PRECEDING** EACH with a SQL*Plus `prompt` command noting what problem part it is for.

2 part a

Write a SQL statement to perform a "pure" relational projection of the client last names from the `client` table.

2 part b

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.

2 part c

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.

2 part d

Write a SQL statement to perform a relational selection of rows of the `rental` table for rentals involving the video with ID of `'00000D'`.

2 part e

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

2 part f

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

2 part g

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

2 part h

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 `hw3-2.sql` file and go into `sqlplus` and see if `start hw3-2.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 `hw3-out.txt` --- at the nrs-labs prompt (the UNIX level, NOT in `sqlplus`!), type:

```
more hw3-out.txt
```

You should see that `hw3-out.txt` contains the query results you just saw within `sqlplus`.

When you are satisfied with these, then `hw3-2.sql` and `hw3-out.txt` are completed.

Problem 3

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

For this part, you will be creating an entity-relationship diagram including entity attribute lists for a scenario.

How are you going to create this? You have several choices:

- you may use Word's or OpenOffice's or NeoOffice's or LibreOffice's drawing tools to draw it,
- you can use other drawing software if you have it,
- you can draw it by hand,
- you can produce part of it using software, and then finish it by hand; etc.

Then, you need to convert your result into PDF format using some means, into a file named `hw3-erd.pdf`;

- you might be using software with an option to save as PDF,
- you can scan it and save it as a PDF,
- you can take a (legible!) photo of it and save it as or convert it to PDF, etc.

You will submit your resulting `hw3-erd.pdf` for this part. (Make sure to put it/them in the same directory as your `.sql` and `.txt` files on `nrs-projects.humboldt.edu`.)

The Scenario:

Consider the following. In a particular club, new members are asked to give their last name and all of their significant e-mail addresses. When accepted as a member, the new member is given a unique membership number, and the date that the member joined the club is recorded, along with their last name and those significant e-mail addresses.

This club offers seminars frequently. To keep track of them, the club gives each seminar a title, a unique seminar number, its date of occurrence, the time that it begins on that date, and the time that it ends on that date. (No seminar lasts more than a day.)

Members may sign up to attend one or more seminars, and do not have to sign up for any. Members may also be in charge of one or more seminars, but do not have to be in charge of any. A seminar must have precisely one member in charge of that seminar, and at least one member must agree to sign up to attend that seminar before it is approved for offering; many members may sign up for each seminar, of course.

Finally, this club has assigned parking spaces in several nearby parking garages. The club has assigned its own unique parking space ID number for each such space; it also keeps track of the name of the garage that that space is in, what section number it is in within that garage, and what space number it is marked with within that section.

A member may be assigned one of these parking spaces, but does not have to be assigned one. No member may have more than one assigned parking space, and a parking space may not be assigned to more than one member at a time (and, of course, some parking spaces may not be assigned to anyone at any given time).

Your Task:

Develop and draw an appropriate entity-relationship diagram including entity attribute lists for the above scenario. Create a PDF of your final entity-relationship diagram including entity attribute lists named `hw3-erd.pdf`, transfer a copy of `hw3-erd.pdf` to `nrs-projects.humboldt.edu`,

and submit it using `~st10/325submit` from there.

Be sure to meet the following style standards:

- each **entity class** is represented by a **rectangle**, with the name of the entity class within the rectangle.
  - Remember: a particular entity class rectangle appears ONLY once within an ERD!
  - (If an entity class is involved in multiple relationships, you just have multiple relationship lines connected to that entity class rectangle.)
- each **relationship** is represented by a diamond on a line connecting the associated entity classes, labeled on or above or below the diamond with a descriptive (and preferably unique) name for that relationship.
  - (that is, you draw a line between related entity classes, and the diamond for the relationship appears on that line…)
- the **maximum** cardinality of each relationship must be depicted by writing the 1, M, or N, whichever is appropriate, near each end of the relationship line near the appropriate entity class rectangle (as depicted in the Reading Packet 4 - Entity-Relationship Modeling, Part 1, available from the course Moodle site, under "CS 325 Reading Packets".)
- the **minimum** cardinality of each relationship class must be depicted by drawing either an oval or a line, whichever is appropriate, on each end of the relationship line (near the entity class rectangle) (as also depicted in the Reading Packet 4 - Entity-Relationship Modeling, Part 1).
  - (Remember: you draw an oval on the relationship line near the entity class rectangle if the relationship is optional at that end, and you draw a line across the relationship line near the entity class rectangle if the relationship is mandatory on that end.)
- remember to include as part of the ERD -- they can be at the bottom, to the side, or on the next page -- the entity attribute lists for each entity class, given as follows: write the name of each entity class, and underneath it put:
  - the attributes for that entity class
  - for any attribute that can be multi-valued, write `(MV)` after that attribute
  - write in all-uppercase the attribute(s), if any, that users in the scenario use to identify/distinguish between different instances of that entity (keeping in mind that **not** all entity classes have identifying attributes)

  Do NOT write these lists of entity class attributes as relation structures -- entity classes are **NOT** relations! Each entity class will eventually be transformed into **one or more** relations during the database design phase, which FOLLOWS the modeling phase.

  Also remember: in the modeling phase, the attributes in these lists are attributes of that entity class alone -- **they do NOT indicate relationships between entity classes**. The relationship lines in the ERD do that!

  A useful rule of thumb: each attribute should only appear once, TOTAL, in this list of all entity class's attributes.
- Including a set of **business rules** is **not required** for this homework problem, unless you would like

to to justify/explain some of your choices (for example, for whether attributes can be multi-valued).

- Note that you may **not** change anything in the scenario, however.

For example, the following meets the above standards:


```
Professor                    Course-section
-----------------            ------------------
PROF_SSN                     Course_name
Prof_lname                   COURSE_ID
Prof_email (MV)              Course_length
```

In the above example, given the maximum and minimum cardinalities shown, a professor does not HAVE to teach any course-sections, but may teach many course-sections. A course-section MUST have an associated professor, and may have at most one professor associated with it -- that is, there is exactly one professor associated with each course-session.

Note that the Professor entity attributes do not include any indication of which courses that professor teaches, nor do the Course-section entity attributes include any indication of which professor teaches that course-section --- this is NOT an error. This is **desired** for the database modeling phase -- the relationship between professors and courses is shown at this phase by the line between their rectangles and by the minimum and maximum cardinalities shown on that line.