CS 325 - Homework 4

Deadline

11:59 pm on **Friday, October 1, 2021**.

Purpose

To get more practice writing SQL `select` statements, and to practice with database modeling (creating an E-R diagram meeting the required notation for this course)

How to submit

Problems 1 and 2 will be completed on the course Canvas site.

Problem 3 will be submitted on nrs-projects.

Problem 4's E-R diagram will be submitted to Canvas.

For Problem 3:

Each time you wish to submit Problem 3, within the directory `325hw4` 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 `4`.

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

Additional notes:

- You are required to use the HSU Oracle `student` database for **Problem 3** of this homework.
- **DB Reading Packet 4** and **SQL Reading Packet 3**, on the course Canvas site, along with the posted in-class projections from the public course web 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 output.
- You are expected to follow **course style standards** for entity-relationship diagrams and SQL `select` statements.

Problem 1

Correctly complete the "HW 4 - Problem 1 - Reading Questions for DB Reading Packet 4 - Entity-Relationship Modeling, Part 1", on the course Canvas site.

Problem 2

Correctly complete the "HW 4 - Problem 2 - Questions about Reading an E-R Diagram", on the course

Canvas site.

Setup for Problem 3

Use `ssh` to connect to `nrs-projects.humboldt.edu`, and create, protect, and go to a directory named `325hw4` on nrs-projects:

```
mkdir 325hw4
chmod 700 325hw4
cd 325hw4
```

Put all of your files for Problem 3 in this directory. (And it is from this directory that you should type `~st10/325submit` each time you would like to submit your files for Problem 3.)

Problem 3

YOU ARE USING ORACLE and SQL FOR THIS PROBLEM.

On the public course web page, along with this assignment, you will find a SQL script `movies-create.sql`. It creates 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
```

(Think of it as a quaint, historical scenario... 8-/)

For your convenience and reference, a handout of these relation structures is posted along with this homework handout.

And, SQL script `movies-pop.sql` initially populates these tables (because it is better style to separate table creation and initial table population).

In your `325hw4` directory on nrs-projects, create a copy of the scripts `movies-create.sql` and `movies-pop.sql`, either by pasting them from the public course web page, or by using these commands at the nrs-projects UNIX prompt while in your directory `325hw4`:

```
cp  ~st10/movies-create.sql  .    # don't forget the space and period!

cp  ~st10/movies-pop.sql     .    # don't forget the space and period!
```

Run these SQL scripts `movies-create.sql` and `movies-pop.sql` in `sqlplus`; these 5 tables should be created, and then initially populated. Look them over; check out their contents.

Then, use `nano` (or `vi` or `emacs`) to create a file named `325hw4.sql` within directory `325hw4`:

```
nano 325hw4.sql
```

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

- your name
- `CS 325 - Homework 4 - Problem 3`
- the date this file was last modified

Then:

- use `spool` to start writing the results for this script's actions into a file `325hw4-out.txt`
- put in a `prompt` command printing `Homework 4 Problem 3`
- put in a `prompt` command printing your name
- 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 `325hw4.sql`, add in SQL statements for the following, **PRECEDING** EACH with a SQL*Plus `prompt` command noting what problem part it is for.

Problem 3-1

Perform a **relational selection** of rows of the `client` table for clients with a client credit rating that is higher than 3.4.

Problem 3-2

Perform a **"pure"/"true" relational projection** of the movie rating and year released columns from the `movie` table.

Problem 3-3

Perform an **equi-join** of the `client` and `movie_category` tables. (Consider: what would be a suitable join condition for an equi-join between these two tables? Which attribute in each has the same domain?)

Problem 3-4

Note that, in this scenario, a rental's date returned attribute is empty/null until the rented video has been returned.

Project just the rented video IDs, date out, and date due for rentals that have **not** yet been returned.

Problem 3-5

Project just the video IDs, video formats, and rental prices for videos that do **not** have the format Blu-Ray.

Problem 3-6

Project just the movie category NAME (**not** the movie category code!), client's last name, and client's credit rating from the equi-join of the `movie_category` and `client` tables.

Problem 3-7

Perform a relational selection of videos with a purchase date between July 15, 2008 and December 1, 2011, inclusive. For full credit, appropriately use the `between` operator in your query.

Problem 3-8

Perform a relational selection of videos which have a rental price greater than or equal to $3.99 and have a purchase date on or after January 1, 2011.

Problem 3-9

Project only the movie title and the movie rating for all movies containing the string `'the'` anywhere in their title (where the case IS significant -- only all-lowercase `'the'` is desired). For full credit, appropriately use the `like` operator in your query.

Problem 3-10

Project only the movie rating and movie title for all movies with ratings of PG-13 or R or A only. For full credit, appropriately use the `in` operator in your query.

Submit your files `325hw4.sql` and `325hw4-out.txt`.

Problem 4 - NOTE! This problem does NOT use Oracle or SQL!

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 `325hw4-erd.pdf`;

- you might be using software with an option to save or export it 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.

NOTE -- because this file `325hw4-erd.pdf` is being created on a computer other than nrs-projects, you will submit it to Canvas.

The Scenario to model:

Consider the following. In a particular club, new members are given a unique membership number, and asked to give their last name and all of their significant e-mail addresses. The date the new member joined the club is recorded, also.

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 `325hw4-erd.pdf`, and submit it to Canvas.

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 DB Reading Packet 4 - Entity-Relationship Modeling, Part 1, available from the course Canvas site, under "CS 325 Reading Packets".)

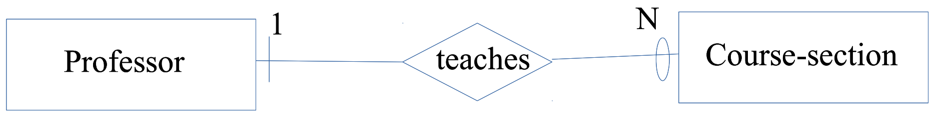
- the **minimum** cardinality of each relationship class must be depicted by drawing either an oval or a line, whichever is appropriate, on each end of the relationship line (near the entity class rectangle) (as also depicted in the Reading Packet 4 - Entity-Relationship Modeling, Part 1).
  - (Remember: 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.