**Assignment 2**

**Purpose:**

**To give you practice:**

* drawing ER diagrams
* converting an ER diagram to a relational database
* creating a database using SQL
* creating tables using SQL
* inserting, modifying and deleting data from tables using SQL
* querying a database using SQL
* doing a bulk load of a table using a comma separated file

**The Problem:**

The database you design should be for the following scenario:  The Western CS Undergrad Chair needs a database to help with academic counseling. He needs one table that just contains Western Computer Science courses. Then he needs to keep track of universities in Canada and he needs a table of all the computer science courses offered by other universities. Finally, he needs to keep track of which Western Course is equivalent to which course from say, University of Toronto and the date that he made the decision about the equivalency.

For our Western computer courses, he needs to keep track of the unique course number (like cs1026, it will always start with cs and then be followed by 4 digits), the course name (like *Introduction to Computer Programming*, up to 50 characters) and the weight (0.5 for half courses, 1.0 for full courses) and a suffix (A/B or F/G  or E or Y or Z or blank (full year course)).  The suffix can be empty but none of the other fields should be allowed to be empty.

For the universities, he needs to keep track of the university's official name (like *University of Toronto*, up to 50 Characters), the city it is in, the province code it is in, it's nickname (e.g. U of T, about 20 characters), and a unique number to identify it (a value from 1 - 99, we will give you this value for each university, do not autogenerate this field). None of the fields should be blank.  The university may or may not have courses it offers that we need to keep track of.

The courses from other universities have codes that start with CompSci and are followed by 3 digits.  Keep in mind that UofT could have a course CompSci020 and Waterloo could have the same named course but they might be totally different courses, the only way to distinguish them is to tell which university they come from. Besides that, the other university courses have a name (up to 50 characters), the year of students it is usually offered for (1,2,3,4) and its weight (0.5 or 1.0)

Finally, you have to keep track of which Western course is equivalent to which other university courses. For example you might have that cs1026 at Western is the same as UofT CompSci030 and that Dr. El Sakka decided this on May 12, 2016 and the same as Waterloo's CompSci032 and Dr. El Sakka decided this on June 11, 2015. The date should NOT be blank.  Some courses may have no equivalent Western course and some Western courses may have no equivalent course from another university. Some outside courses may be equivalent to more than one Western course.

In this example, because the Western courses store more and different information than the other universities courses, do NOT put them in the same table and do not use generalization or specialization for this situation. Just have one entity represent the Western courses and another entity (think about the type of entity it should be!) represent the other universities courses (Maybe call this one OutsideCourse).

**Instructions:**

There are 2 parts for this assignment. In part 1 you will draw an ER diagram and in part 2 you will convert your ER diagram into a database and  create the database on your virtual machine using MySQL

**Part 1:**

For the above scenario, draw an ER diagram using MS Visio or draw.io.   Then save your ER diagram as yourwesternuseridERDiagram.jpg (or .pdf, or .gif). You should show the total or partial participation using the single/double lines. You do NOT need to include the (min,max) this time. Do not use Crows Feet Notation.  Make sure you indicate the primary key and the cardinality.

**Part 2:**

Using your ER diagram from above, figure out what tables you will need for your database. Create the following 4 script files (more details to follow):

1. File 1: Delete the database and create a new database  and create the tables
2. File 2: Insert some data and update the data
3. File 3: Query the data
4. File 4: Delete some data, add a constraint, add a view

Make sure the SQL commands are UPPER case and the table names and column names are lowercase.

**IMPORTANT:**

Get onto your virtual machine using your terminal window or putty (use vm???@cs3319.gaul.csd.uwo.ca, then ssh centos@vm??? where ??? is the number for the virtual machine you were assigned) and do the following:

* Make sure that you have set up your Bit Bucket Account properly. This video walks you through it and helps you set it up so that you don't have to type your password each time you want to push or pull files -  it shows you how to set a public/private key pair on both your laptop and on your vitual machine (This is shown starting at Minute 0, Second 45). It also shows you how to set up your directory for assignment 2 on your VM and push from the VM to Bit Bucket.(This is shown starting at about Minute 7, Second 10). You probably want to watch the whole 20 minutes and follow along on your computer.  Here is the link: <https://www.csd.uwo.ca/~lreid2/blendedcs3319/videolectures/week5/video2019/bitbucketpart3/bitbucketpart3.html>
* From your home directory, create a directory called ***assignment2***as shown in the video (this step is around Minute 13):
* Make sure that you create your 4 script files for the assignment (see below for description of the script files) in that ***assignment2***directory that is now in Bit Bucket.
* Make sure that you FREQUENTLY push your script files up to Bit Bucket. The t.a. may check Bit Bucket to see your work and time stamps so MAKE SURE YOU PUSH YOUR SCRIPT FILES OFTEN!

**File 1 - Creation of Database and Tables**

Create a script file that contains the MySQL commands to:

* List all the current databases owned by you
* Delete the database called yourwesternuseridassign2db if it exists
* Create a database called yourwesternuseridassign2db
* Connect to (use) that database
* Add these lines to make sure the t.a. has access to your database in order to mark this assignment:  
  GRANT USAGE ON \*.\* TO 'ta'@'localhost';  
  DROP USER 'ta'@'localhost';  
  CREATE USER 'ta'@'localhost' IDENTIFIED BY 'cs3319';  
  GRANT ALL PRIVILEGES ON yourwesternuseridassign2db.\* TO 'ta'@'localhost';  
  FLUSH PRIVILEGES;
* List all the tables (should be none initially)
* Create the tables you need to solve the problem above with the appropriate types and keys, foreign keys.  Make sure that:
  + you create the foreign keys (this may force you to create the tables in a certain order in order to indicate the foreign keys)
  + when you delete a university, cause an error to occur if the university still has any courses offered by it in the outside courses table and  don't let the university be deleted. If you delete a Western Course that is equivalent to an outside course OR you delete an outside course that is equivalent to a Western Course, make sure you also delete those equivalencies (HINT: Google CASCADE for the delete command)
* List the tables again
* Call this script: *yourwesternuseridscript1.sql*
* After you have the script working, save the output from it using the following command (DO NOT copy and paste this command, type it in yourself because we have had issues when students copy and paste from owl):  
      **s*udo mysql -nvvf --verbose -pyourpassword < yourwesternuseridscript1.sql  > yourwesternuseridoutputscript1.txt 2>&*1**
  + NOTES:
    - make sure you turn -pyourpassword to be whatever your MySQL password is. Likely you will put: ***-pcs3319***
    - < means pipe everything in the usernamescript1.sql INTO mysql to execute
    - > means pipe all the output from the commands in the input script to usernameoutscript.txt
    - 2>&1 means also pipe (>) any errors (errors go to 2) into the same location as the stdout(&1), theoutput script file as well rather than to the terminal

**File 2 - Insertion and Modification**

* HINT: You will need to make sure you are connected to the correct database as the first step in this script.
* REMEMBER that before you start your inserts, do a SELECT command to prove the table was empty AND after you insert the all the data for each table, do a SELECT to prove that it now has data.
* Using the***LOAD DATA LOCAL INFILE ...*** mysql command (you will need to Google to see the syntax for this command), load the following data into the ***patient***table:  [[http://www.csd.uwo.ca/~lreid/blendedcs3319/assignments/assignment2/loaddatafall2020.sql](http://www.csd.uwo.ca/~lreid/blendedcs3319/assignments/assignment2/loaddatafall2020.txt)]
  + NOTE: You will need to make a copy of the above data file in your assignment2 directory on your virtual machine. You cannot directly link to the above file.
* Using the SQL INSERT statement, insert the following data also into the tables
  + **Western Courses:**
    - cs1026, Computer Science Fundamentals I, 0.5, "A/B"
    - cs1027, Computer Science Fundamentals II, 0.5, "A/B"
    - cs2210, Algorithms and Data Structures, 1.0, "A/B"
    - cs3319, Databases I, 0.5, , "A/B"
    - cs2120, Modern Survival Skills I: Coding Essentials, 0.5, , "A/B"
    - cs4490, Thesis, 0.5, "Z"
    - cs0020, Intro to Coding using Pascal and Fortran, 1.0, ""  (this is the course Laura took when she was in first year!)
    - a course that you wished our department offered!
  + **University:**
    - Add one more university, a fictitious one, somewhere in Canada!
  + **Outside Courses:**
    - Courses at UofT:
      * CompSci022, Introduction to Programming, 1, 0.5
      * CompSci033, Intro to Programming for Med students, 3, 0.5
      * CompSci021, Packages, 1, 0.5
      * CompSci222, Introduction to Databases, 2, 1.0
      * CompSci023, Advanced Programming, 1, 0.5
    - Courses at Waterloo:
      * CompSci011, Intro to Computer Science, 2, 0.5
      * CompSci123, Using UNIX, 2, 0.5
    - Courses at UBC:
      * CompSci021, Intro Programming, 1, 1.0
      * CompSci022, Advanced Programming, 1, 0.5
      * CompSci319, Using Databases, 3, 0.5
    - Courses at Mac:
      * CompSci333, Graphics, 3, 0.5
      * CompSci444, Networks, 4, 0.5
    - Courses at Laurier:
      * CompSci022, Using Packages, 1, 0.5
      * CompSci101, Introduction to Using Data Structures, 2, 0.5
    - Courses at Your University:
      * Make up two courses for your university
    - **The undergraduate chair has decided to allow the following equivalencies:**

|  |  |  |  |
| --- | --- | --- | --- |
| **Western Course** | **OutsideCourse** | **University** | **Date This Was Decided:** |
| cs1026 | CompSci022 | UofT | May 12, 2015 |
| cs1026 | CompSci033 | UofT | Jan 2, 2013 |
| cs1026 | CompSci011 | Waterloo | Feb  9, 1997 |
| cs1026 | CompSci021 | UBC | Jan 12, 2010 |
| cs1027 | CompSci023 | UofT | Jun 22, 2017 |
| cs1027 | CompSci022 | UBC | Sep 1, 2019 |
| cs2210 | CompSci101 | Laurier | Jul 12, 1998 |
| cs3319 | CompSci222 | UofT | Sep 13, 1990 |
| cs3319 | CompSci319 | UBC | Sep 21, 1987 |
| cs2120 | CompSci022 | UofT | Dec 10, 2018 |
| cs0020 | CompSci022 | UofT | Sep 17, 1999 |

* Make sure you remembered to show what was in the table before and after in each table to prove that your inserts worked.
* Using the SQL UPDATE command, update the data as follows (remember to show the table BEFORE and AFTER you do each of the updates below):
  + Change the date that cs0020 was evaluated to be the same as CompSci022 at UofT to be September 19, 2018
  + Change the year to be 1 for every course from other universities that has a course name that starts with the 5 characters "Intro".
* Call this script: *yourwesternuseridscript2.sql*
* After you have the script working, save the output from it using the following command:

**s*udo mysql -nvvf --verbose -pyourpassword < yourwesternuseridscript2.sql  > yourwesternuseridoutputscript2.txt 2>&*1**

**File 3 - Queries**

Write SQL commands that query the data to:

1. Show the course names of all the Western Courses
2. Show the course numbers of all courses from other universities with no repeats
3. Show all the data in the Western Course table, but show them in order of their course names from A to Z;
4. List the course number and title of all half (0.5) Western courses.
5. List all the course numbers and course names from courses offered at the University of Toronto (you must use the University's Name as the lookup in the query not the unique university number)
6. List the outside university course name and the university nickname of any course whose name has the word "Introduction" somewhere in the name.
7. List all courses names, their universities names, the Western course name, the evaluated date of equivalent courses but only those courses that have not been reevaluated in the last 5 years. (Google the MySQL command's CURDATE() and DATE\_SUB() with INTERVAL YEAR to help you make this work no matter what date it is fun).
8. List the all courses names and the university nicknames of course that are equivalent to Western's cs1026.
9. Count the number of courses that are equivalent to Western's cs1026
10. List all Western course names and the outside course name and the university nickname  of Western courses that are equivalent to a course offered by ANY university in Waterloo, Ontario.
11. List all University Names where they do not have any courses equivalent to a Western Course.
12. Find the course name of any Western course and Western course number that is equivalent to a third or fourth year class at another university.
13. Find the names of any Western courses that are equivalent to more than one outside course. (Hint: you will have to use the key words *Group By* and *Having*)
14. We want to make sure that students dont get credit for a half course from another university that was supposed to be a full course at Western or vise versa. Write a query that finds all outside courses that are equivalent to one or more of the Western courses but dont have the same weight as that course.  Output the Western course num and name and weight and output the outside course name and university name and weight.  Write the query so that the columns have the following neat titles "Western Course Number", "Western Course Name", "Course Weight", "Other University Course Name", "University Name" and "Other Course Weight".

* Before each of the above queries, put an MySQL  comment that tells the query number to help the t.a.s figure out which queries are answering each of the questions above. . In MySQL comments start with -- , so do something like:  
  --  Query 1  
  SELECT \* FROM ...
  + NOTE: when doing comments in script files you MUST put two hyphens and then you MUST put a space, if you put 3 or more hyphens in a row or 2 hyphens with no space, the comment does NOT work. For example:
    - These 2 comment lines will NOT work:  
      **-----------  
      --Query 1  
      SELECT \* FROM ...**
    - These 2 comment lines WILL work:  
      **-- ---------  
      -- Query 1  
      SELECT \* FROM ...**
* Call this script: *yourwesternuseridscript3.sql*
* After you have the script working, save the output from it using the following command:

**s*udo mysql -nvvf --verbose -pyourpassword < yourwesternuseridscript3.sql  > yourwesternuseridoutputscript3.txt 2>&*1**

**File 4 - Deletions/Views**

* Create a view that shows any course title from another university, the university name, nickname and city that it is from and the Western course name that it equivalent to, but only for courses for year 1 students. Then show all the records in this view.
* Prove that your view works by selecting all the rows from it.
* Run your view again but this time just select all the columns from the view for universities with the nickname "UofT" . Order the rows in ascending order by the Western Course name.
* Show all the university table information
* Delete any university that has a nickname with the letters "cord" in it.
* Show all the university table information again to make sure it was remove
* Delete any university from Ontario
* Put a comment (-- the reason why ...) in your script file to explain why the Ontario universities could not be delete
* Show everything in the university table again
* Show all the information about the outside course and all the information the university they are associated with
* Delete all the courses that are offered from a university in the city of Waterloo. Make sure you use city = "Waterloo" in your delete statement.
* Show all the information about the outside course and all the information the university they are associated with to make sure those courses were deleted.
* Run your view again and make sure that the equivalencies were also deleted from the view (Just double check that the rows returned from the steps above where you create your view are less now)*.*
* Call this script: *yourwesternuseridscript4.sql*
* After you have the script working, save the output from it using the following command:

**s*udo mysql -nvvf --verbose -pyourpassword < yourwesternuseridscript4.sql  > yourwesternuseridoutputscript4.txt 2>&*1**

**Notes:**

* Here is a sample script file:  [[http://www.csd.uwo.ca/~lreid/cs3319/assignments/assignment2/samplescriptmysql.sql](http://www.csd.uwo.ca/~lreid/cs3319/assignments/assignment2/samplescriptmysql.txt)]
* Do the steps (inserts, deletes, etc...) in the order given
* Put MySQL comments (comments start with -- ) anywhere that you think might help the t.a. figure out what you are doing.
* For any insert, delete or update commands remember to show the table before and after you modified it
* Remember: Best practice is for SQL keywords to be uppercase and the tables and columns to be lowercase (or camel case)

**Handing in the Assignment**

You are required to submit the following files via Owl:

* your ER diagram (named yourwesternidERDiagram.jpg/.pdf/.gif)
* your 4 script files
* your 4 files containing the output from the scripts.

**HINT:**

To make your scripts, the easiest way to do it is to create a .sql file on your local machine (in Notepad for example) with the commands in it, test those commands by copying them from the .sql file and then pasting them into your virtual machine window, make any necessary changes for commands that don't work in the Notepad .sql file, then save the .sql file every so often.   When you are ready to do your final version, copy the code from the .sql local file and paste it into the virtual machine window (and occasionally use Push with Bit Bucket). **THIS METHOD WILL AVOID TYPING ERRORS AND LOSING YOUR FILES!!!**