To submit:
Write up -> pdf !!
readme.txt
makefile
load.cpp
query.cpp
maybe extra credit

ECS165A FQ16

November 17, 2016

To do:
看alan write up，加update method
alan写update conflict courses term
extra credit
(update alan 3e)

Homework 4

Due 11:59PM December 2, 2016. **Only PDF** will be accepted for the write up. **No scans** of handwritten work will be accepted. For this homework you will be working in groups of two, a group of three will only be allowed with approval. All programs will be evaluated on the CSIF. Upload your files as a tar gzip file (tgz). This specification is subject to change.

CID,TERM,SUBJ,CRSE,SEC,UNITS always non-empty for valid courses

You are designing a database for a university called FakeU. As a trial you have been provided grade data from courses for departments ABC and DEF. The grade data is from Summer of 1989 until Summer of 2012. The data provided is in CSV format, and is only as complete as could be made possible. There may be errors, omissions or redundant data in the files. FakeU like UC Davis is on a quarter system, however they have recently transitioned to a single summer quarter instead of two summer sessions. This has corrupted some of their summer data as all summer session classes have now been grouped into a single summer quarter term. Each course has a course ID (CID), a term it was offered (TERM), a subject (SUBJ), a course number (CRSE), a section (SEC), and number of units (UNITS). Within a course there listings of meetings, the instructor of the meeting (INSTRUCTOR(S)), meeting type (TYPE), day of meeting (DAYS), time of meeting (TIME), meeting building (BUILD), and meeting room (ROOM) are also listed. For each student that takes the course there is a student seat (SEAT), a student ID (SID), the student's surname (SURNAME), the student's preferred name (PREFNAME), the student's (LEVEL), the number of units the student is receiving (UNITS), the student's class standing (CLASS), the student's major (MAJOR), the grade the student received in the course (GRADE), the student's registration status (STATUS), and the student's e-mail address (EMAIL).

1) Create a database schema for the grade data. Describe your schema in your write up; specifically describe how you expect to be able to update your database with future updates of the data. Make sure you describe how you will store the instructor, student, building, course, etc. information. You must list the functional (and multivalue) dependencies that you expect to hold for each relation.

2) Write a program to load the grade data into a PostgreSQL database that follows your schema. Your program can be written in C++ or python, you may **NOT** use standalone SQL files. If you choose to make a C++ program, you must include a makefile. Include a readme file with directions of how to use your program. If you choose to make a python program you must specify which version of python you used.

3) Write another program to query your database to calculate the following values, put the results in your write up, some may be best described with a chart instead of raw values:

    a) Calculate the percent of students that attempt 1 – 20 units of ABC or DEF per quarter for every unit increment (e.g. 1, 2, 3,…).

group by SID and TERM, sum units -> studentUnit

avg(studentUnit = 1…20 in sid-term/sum student in sid-term)

possible that 2 courses have same CID, TERM, SUBJ, CRSE;
in the future, CID, TERM for key

1) find SID take 1…20 units (group by Term+SID, sum units)
2) 20 values
2) 20 values

b) Calculate the average GPA for the students that take each number of units from part a. Assume that the grades have standard grade points (A+ = 4.0, A = 4.0, A- = 3.7, B+ = 3.3…).

1) InstrC natualJoin StudC,
2) sum up GRADES,
summer meraging doesnt matter

NOTE: null instructors

1) InstrC natualJoin StudC,
2) sum up GRADES,
summer meraging doesnt matter

c) Find the easiest and hardest instructors based upon the grades of all the students they have taught in their courses. Provide their name and the average grade they assigned.

d) Do the analysis again for each ABC 100 level course that is offered, which instructor is the easiest/most difficult. For each course provide the instructors name and the average grade they assigned.

e) Find the list of courses that must have been offered in different summer sessions as they have meeting conflicts. Only list the pair once, put the course name/number string in alphabetically order of the pairs.

possible that 2 different courses have same CID, TERM, SUBJ, CRSE;
in the future, CID, TERM for key

f) What major performs the best/worst on average in ABC courses?

see term and major

g) What percent of students transfer into one of the ABC majors? What are the top 5 majors that students transfer from into ABC, and what is the percent of students from each of those majors?

4) Extra credit: The Efficient XML Interchange (EXI) is a format for the compact representation of XML information. The CSV files provided for this assignment have been consolidated into a single EXI file (HW4Grades.exi) that is available in the resources section of Canvas. Implement a separate program that it can load the database from the EXI file. You may **NOT** use shell calls, or creation of external temporary files for this part.

5) Extra credit: Additional queries/query program.

a) What courses appear to be prerequisites for ABC 203, ABC 210, and ABC 222? For this problem list the courses that the X% of students have taken for every 5% increment from 75% - 100% prior to taking the course.

b) Write a program that will find an open room for course expansion. The program must prompt for term, CID, and number students to add. The room(s) returned should be ordered from best to worst fit with up to 5 results. Assume that each room capacity is the maximum number of students listed for any particular meeting in the data files (don't forget that lectures may be split across multiple CIDs).

Some useful tips:

- When loading the tuples into the database, insert them in batches. Inserting one tuple at a time may cause the program to take on the order of tens of minutes or hours instead of a few minutes.
- Test a subset of the data first.