CSci 4707: Practice of Database Systems
Lab 1
Spring 2017

Due: 02/28/2017 18:30 on Moodle
There will be a 10% penalty off your grade for a 24 hours late submission.
This lab must be done **individually**.

A. Introduction

Each student has to do this project individually. You are going to use PostgreSQL, a major open source DMBS, to perform some queries against a predefined database. The schema and sample data of the database are provided. The project should be able to run on any CSE Linux machines.

B. Lab Overview

In this project, you will create a ProjectsDB database. ProjectsDB is a database used by a software development company to keep track of its projects. ProjectsDB keeps track of the projects, the employees/managers working on the projects including the universities they were graduated from. In particular, in this lab, you will perform the following tasks:

- Setup the PostgreSQL database.
- Create the schema of the ProjectsDB database.
- Perform multiple queries.

C. Setting-Up PostgreSQL

PostgreSQL is a large software. You will want at least 200 MB of space in your machine. Download version 9.6.1 of PostgreSQL and create the installation location by executing the following commands:

```
// Download the PostgreSQL and extract it.
wget https://ftp.postgresql.org/pub/source/v9.6.1/postgresql-9.6.1.tar.gz
tar xvzf postgresql-9.6.1.tar.gz

// Create the installation folder, called “install”.
mkdir install
```

Then, we retrieve the full path of both source and installation folder by executing the following command:

```
// Retrieve the full path of the directory.
pwd
# This will return a full path of your current directory. Throughout this
documentation, this path is referred as $W$. So, Don't forget to replace it with
your actual working directory.
```

Thus, the path of the PostgreSQL's source code is located at "`$W$/postgresql-9.6.1`" and the path where we will install PostgreSQL is located at "`$W$/install`".

Next, you will configure the installation of PostgreSQL and install PostgreSQL on the given path by executing the following commands:

```
// Go to the source directory.
cd postgresql-9.6.1

// Run the configure command. Don't forget to change $W$ to your actual
directory.
./configure --prefix=$W$/install

// Run the makefile.
make
# You should see the following output.
All of PostgreSQL successfully made. Ready to install.

// Run the installation.
make install
# You should see the following output.
PostgreSQL installation complete.
```

After the installation has finished, you will need to initialize and create a database to use. In this example, we will initialize the database at "`$W$/install/data`" folder by executing the following command:

```
// Go to the installation folder.
cd ../install
// Initialize database.
bin/initdb -D data
```

Once the database has been initialized, we can start the PostgreSQL backend by executing the following command:

```
// Execute the backend.
bin/postgres -D data
```

We can then use this terminal to get information on what is happening in our DBMS.

To run the client, open another terminal and use the following command to create a database to use: (In this example, we call the database as ProjectDB)

```
// Create the ProjectDB database.
cd $W$/install
bin/createdb -h localhost ProjectDB
```

We interact with our DBMS by using a program called psql. To start psql, use the following command:

```
// Run psql
bin/psql -h localhost ProjectDB
```

The next time you want to start the database again, you only need to run two consoles where one is for the backend and one is for the psql console:

```
// Execute the backend.
bin/postgres -D data

// Execute the psql console.
bin/psql -h localhost ProjectDB
```

Several important commands for the lab are:

```
// You must exit psql by typing the following command and do not use Ctrl+C
\q

// To run an sql script located at $W$/script.sql
\i $W$/script.sql
```

D. PART 1: ProjectDB Schema Creation (20 Points)

For the syntax of each command in psql, please consult the documentation of PostgreSQL here.
ProjectDB consists of the following relations defined in the following schema:

University (
    UnivId: NUMERIC
    UnivName: VARCHAR(40)
)
Department (
    DeptId: NUMERIC
    DeptName: VARCHAR(40)
)
Employee (
    EmpId: NUMERIC
    EmpName: VARCHAR(40)
    DeptId: NUMERIC REFERENCES Department(DeptId)
    HomeZipCode: NUMERIC
)
Project (
    ProjId: NUMERIC
    ProjName: VARCHAR(40)
)
Graduate (
    EmpId: NUMERIC REFERENCES Employee(EmpId)
    UnivId: NUMERIC REFERENCES University(UnivId)
    GradYear: NUMERIC
)
EmpProject (
    EmpId: NUMERIC REFERENCES Employee(EmpId)
    ProjId: NUMERIC REFERENCES Project(ProjId)
)
ProjectManager (
    ProjId: NUMERIC REFERENCES Project(ProjId)
    MgrId: NUMERIC REFERENCES Employee(EmpId)
)

What to Submit?

The result of your work is a file named p1_<x500>.sql file, where <x500> is your X500 id, that contains all SQL queries to create the above relations. Your script must be able to be ran on any CSE machines and successfully create the above relations.

Grading Criteria:

- 10 Points: The provided script is able to create all the above relations with the correct schema.
- 10 Points: The provided script handles the FOREIGN KEY constraint perfectly.

E. PART 2: SQL Queries (60 Points)

Once you have created the above schema, run the “`data.sql`” file to load the data into the database. Then, write SQL queries that answer the questions below (one SQL query per question but you are allowed to use nested queries). The query answers should be duplicate-free, but you should use distinct only when necessary.

1. Find the names of the employees who are living in Minneapolis (Zip code 55414 or 55455).
2. Find the names of the projects that are currently managed by any manager.
3. For each project, display its name as well as the number of employees who are currently working on it except for a project without any employees.
4. Find the name(s) of the university/universities that graduated the maximum number of distinct managers.
5. For each employee, say E, display the name of E, the department name of E, and the graduation year of E.
6. Display the name of the project that has the maximum number of different employees who worked/”are working” on it. If more than one project qualify, display all the qualified projects. (Hint: refer to EmpProject only and not ProjectManager).

What to Submit?

The result of your work is a file named p2_<x500>.sql file, where <x500> is your X500 id, that contains all the above SQL queries.

Grading Criteria:

- 10 Points (no partial credit) for each correct SQL query.

F. PART 3: SQL Updates and Deletes (20 Points)

1. Employee with Id 2 has moved to an address in 55414. Please update this in the database and select that employee after the update to ensure that the corresponding tuple has been updated.
2. Just as a practical joke, increment the graduation year of every graduate who graduated before 2002 by three.

3. Just another practical joke, decrement the graduation year of every graduate who lives in 55414 by two.
4. For some reasons, the database owner wants to get rid of everything related to the project with Id 2. Please delete all the tuples related to Project 2 (as if that project has never existed before).

What to Submit?

The result of your work is a file named p3_<x500>.sql file, where <x500> is your X500 id, that contains all the above SQL queries.

Grading Criteria:

- 5 Points (no partial credit) for each correct SQL query.

G. Submission Guideline

In addition to the three sql files above, provide a readme file, named README.txt which contains your name, your student Id, and your X500. Zip all four files (three sql files and one readme file) into one zip file named lab1.zip and submit the zip file to Moodle. By not following the submission guideline, including the naming of each file, will result in 10% penalty for each violation.