CS 361 – Phase 3 OIE Project – SQL Queries and Data Manipulation
Due: Submitted to D2L dropbox in format described below by Friday, April 14

In this phase of your project, you will (1) load sample test data into each of the tables of your OIE database and (2) then design queries or data manipulation statements which begin to address the use cases that Jenna Graff has provided for us.

Loading Sample Test Data (32 points)

Each of the tables in your database should be loaded in a way that would give Jenna a reasonable sampling of at least 50 students, 5 Hessen institutions, and 5 UWSystem institutions to explore. Each institution should have at least three staff members in different roles. The data should be "fake" but resemble real data. That is, phone numbers should resemble actual phone numbers, email addresses should resemble actual email addresses, etc.

A recommended way of doing this is to download assorted CSV spreadsheets of fake data from `http://www.fakenamegenerator.com/order.php`, manipulate those spreadsheets as you need in Excel or LibreOffice, and then use the bulk-loading technique described below.

Bulk loading data in MySql

The mysqlimport command can be conveniently used to bulk load data from a CSV file to a table using the following process:

- Suppose that you have a DB called *bulk_load_example* with a table called *some_data*.

```
mysql> describe some_data;
+--------------+-------------+------+-----+---------+-------+
| Field        | Type        | Null | Key | Default | Extra |
+--------------+-------------+------+-----+---------+-------+
| Title        | varchar(5)  | YES  |     | NULL    |       |
| GivenName    | varchar(32) | YES  |     | NULL    |       |
| SurName      | varchar(32) | YES  |     | NULL    |       |
| EmailAddress | varchar(60) | YES  |     | NULL    |       |
+--------------+-------------+------+-----+---------+-------+
4 rows in set (0.05 sec)

mysql> select * from some_data;
Empty set (0.05 sec)
```

- Suppose also that you have a CSV file *some_data.csv* with some fake data like that I downloaded from `http://www.fakenamegenerator.com/order.php`

```
Title,GivenName,Surname,EmailAddress
Mr.,Lucas,Church,LucasChurch@einrot.com
Dr.,Jai,Foster,JaiFoster@superrito.com
⋮
Ms.,Charlotte,McEwan,CharlotteMcEwan@gustr.com
Mr.,Jacob,Jervois,JacobJervois@fleckens.hu
Ms.,Jade,Adey,JadeAdey@superrito.com
Mr.,Aaron,Sennitt,AaronSennitt@gustr.com
Mrs.,Jessica,Gabriel,JessicaGabriel@superrito.com
```

- To bulk load this data, issue the following command:

```
$ mysqlimport -u username -h labdb.acs.uwosh.edu -p'password' --fields-terminated-by=","
      --ignore-lines=1 --local bulk_load_example some_data.csv
```

```
The above should be typed on one line at your command prompt.
The name of the CSV file should match the name of the table in the DB.
MySql would respond with:

bulk_load_example.some_data: Records: 30  Deleted: 0  Skipped: 0  Warnings: 0
```

- Consequently you will find that your data has been loaded from the CSV file into your table:

```
mysql> select * from some_data;
select * from some_data;
+----------+-----------+-------------+--------------------------------+
| Title    | GivenName | SurName     | EmailAddress                   |
+----------+-----------+-------------+--------------------------------+
| Mr.      | Lucas     | Church      | LucasChurch@einrot.com         |
| Dr.      | Jai       | Foster      | JaiFoster@superrito.com        |
⋮
| Ms.      | Charlotte | McEwan      | CharlotteMcEwan@gustr.com      |
| Mr.      | Jacob     | Jervois     | JacobJervois@fleckens.hu       |
| Ms.      | Jade      | Adey        | JadeAdey@superrito.com         |
| Mr.      | Aaron     | Sennitt     | AaronSennitt@gustr.com         |
| Mrs.     | Jessica   | Gabriel     | JessicaGabriel@superrito.com   |
+----------+-----------+-------------+--------------------------------+
31 rows in set (0.05 sec)
```

- The *fakenamegenerator* site also has an option to download your fake data in the form of SQL INSERT statements that can consequently be edited and then executed in MySqlWorkbench if that proves more convenient for your way of working.

Once you've got your large tables loaded with sample data, you will need to get data into those tables that reflect relationships from your ER model. Typically these tables have fewer attributes with composite keys comprised of a pair of attributes, each of which comes from a different table. To automate the process of creating these tables, use the strategy of employing an INSERT of the form: `INSERT INTO <relation> ( <subquery> );`
See the slides of March 13 for details.

The fake data you load should be designed to showcase the use cases below for which you choose to develop queries or data manipulation statements.

Design of queries or data manipulation statements (68 points)

Pick 17 of the 21 use cases at the end of this document. Then for each of those, design a strategic query or data manipulation statement that suitably illustrates how a suite of similarly parameterized queries and data manipulation statements could ultimately provide everything Jenna needs for that use case. The illustrative query you design for each use case must not be a trivial query like

```
SELECT * FROM students;
```

It can have hard-coded values in its WHERE or ON clauses. That is, the data on which selections are being made can be constants instead of parameters whose values are coming from a web front-end connecting to the database. In this phase you are not designing the entire suite of queries for any particular use case. You are merely demonstrating that your database has progressed to the point where it can begin addressing most of Jenna's needs. In the final phase of the OIE project you will flesh out a complete suite of parameterized queries for a smaller number of use-cases, along with a modest GUI that can be employed by a user to provide values for the query to work with.

What should you submit for your work on Phase 3?

Submit a (text) file named *phase3.sql* to the D2L dropbox for Phase 3. That SQL file should have, for each use case you work on, a section in *exactly* the following format:

```
use XXXXXX;                                                    -- XXXXXX is the name of your database

-- Query for use case N                                (N is the number of the use case from list at end)

-- An explanation of how your query fits the use case

select '***' from dual;
select 'Use Case 1' from dual;
select '***' from dual;

Your query would appear here
```

For example, if Aaron Rodgers were illustrating a query for Use Case 1, the corresponding section in his sql file would look like:

```
use rodgersa;

-- Query for use case  1
-- The query below illustrates how a staff member could display names and
-- email addresses of all Hessen students, grouped by their home Hessen
-- institution, along with their first and second choices for a UW System campus.

select '***' from dual;
select 'Use Case 1' from dual;
select '***' from dual;

SELECT name, emailAddress, UW1, UW2 FROM HessenStudents
       GROUP BY HessenInstitution;
```

The SELECT statement above is based on an imagined schema. Obviously the particular SELECT statement you design for Use Case 1 would be dependent on your schema and very different from the illustration above.

Be forewarned that I will run your sql file from the command line exactly like I did in Assignment 3, that is:

```
$ mysql -u username -h labdb.acs.uwosh.edu -ppassword < phase3.sql
```

I should see the output from the duals, plus MySql's response for each of your 20 queries. If it crashes, you won't receive credit for any of the use cases after the crash, so you will definitely want to test this file from the command line before you submit it.

Whether or not you get the full point value for any particular use case will depend on your query or data manipulation statement working successfully and the quality of your commented explanation as to how your query fits the use case.

Use cases

Here are the 32 use cases, which essentially comprise Jenna's original list augmented by a few additional cases that grew out of her in-class discussion with us.

1. The names and email addresses of all Hessen students, grouped by their home Hessen institution. Also display their first and second choices for a UW System campus.

2. The names and email addresses of all UW System students, grouped by their home UW System institution. Also display their first and second choices for a Hessen campus.

3. The names and email addresses of all UW System students, grouped by the term for which they are applying to study at a Hessen campus.
4. For a given student, display their requested courses.
5. The names and roles of each UW System staff member, grouped by the UW System campus they represent.
6. For a given UW System advisor, the names of the students working with that advisor.
7. Names and contact information regarding all outbound student applications from a particular UW System or Hessen campus grouped by those students whose application materials are complete and those whose materials are not complete.
8. Names and contact information regarding all outbound student applications from a particular UW System or Hessen campus that are not yet complete, along with an indication of the application materials that still have to be completed for each student.
9. A group of three queries that:
   (a) Display the name of a student whose application materials are not yet complete
   (b) Update the student's record indicating that those materials are now complete
   (c) Display the updated record
10. A group of three queries that:
    (a) Display the name of a student who has not yet been accepted
    (b) Update the student's record indicating that their top choice for an institution has accepted them
    (c) Display the updated record
11. For a particular student who has been accepted, display the date, time, and user associated with each of the student's transitions from the application received state, to the application complete state, to the application accepted state.
12. For a given year, for each host institution, monitor the number of seats remaining (quota) as compared to the number of student applicants for that institution.
13. Display the names of all UW System students whose application to a Hessen institution was declined. Group the students by their UW System campus.
14. For a given term display the Bachelor's and Master's headcounts of Hessen students at each UWSystem campus.
15. For a given term display the Bachelor's and Master's headcounts of UWSystem students at each Hessen campus.
16. For a given term display the Bachelor's and Master's FTE of Hessen students at each UWSystem campus.
17. For a given term display the Bachelor's and Master's FTE of UWSystem students at each Hessen campus.
18. For a given Hessen student who applied and then withdrew their application, provide the following group of three queries:
    (a) Display their record before the application was withdrawn
    (b) Update the record to indicate the application was withdrawn

(c) Display their record after the application was withdrawn

19. Repeat the previous use case, except this time for UW System students who withdraw their application.

20. For a given UWSystem institution, change the contact information of one of that institution's coordinators. To do this, provide the following group of three queries:

    (a) Display the institution name and coordinator contact before the change

    (b) Update the record

    (c) Display their record after the update

21. For a given UWSystem institution, change their application fee or orientation fee. To do this, provide the following group of three queries:

    (a) Display the institution name and fee

    (b) Update the record

    (c) Display the record with the new fee after the update