CS 2243 (Database Management Systems)
Fall 2018

Group Project
Due: Nov. 18, 2018

This project gives you practice at building and using relational databases in SQL.

SQL: You should refer to its online documentation, and you are suggested to use **MySQL** relational database management systems.

The Problem

Suppose a university – let it be Lipscomb University (LIPSCOMB) - is using Microsoft Excel to record information about its students, the classes they take during one semester, and the professors who teach the classes. LIPSCOMB users now realize that they cannot do many of the queries they want in the Excel spreadsheets and they are also worried that the data entry is introducing errors and inconsistencies that the spreadsheets don't check for. They now want you to convert the data into a well-defined relational database and generate some standard queries.

The data for the LIPSCOMB student registration database is as follows:

- Student name (last, first, MI), address (street, city, state, zip), telephone number, class (FR, SO, JR, SR), date of birth, PIN (personal identification number), advisor and date enrolled
- Course number (such as IT 220), course name, section (number, days, time), credits, location (building code, room, capacity), duration, maximum enrollment, instructor, and term offered (term description, status (open/closed), start date)
- Instructor (Faculty) name (last, first, MI), office location, telephone number, rank, supervisor, and PIN
- Student enrollment and grade information

They now want you to convert the data into a well-defined relational database and generate some standard queries.

You will need to:

- Define the schema of the **LIPSCOMB_Student_Registration** database using SQL,
- Populate the schema using the data files. This will require some transformation of the data files, and probably making some temporary files or tables.
- Check that the database enforces the appropriate consistency checks by performing a series of appropriate updates that should be rejected by the database.
- Construct a set of queries for the database.

The Data:

The data files, in the attached handout, contain all the information required:

- location.data: identifies building codes for lectures, room numbers, and room capacities.

- faculty.data: shows faculty members. The first record shows faculty Teresa Marx, whose office is located in BUS 424, and whose telephone number is 325-123-4567. She has the rank of Associate (Associate Professor), and her supervisor is Jonnel Brown.

- student.data: shows student records. The first record displays data for student Tammy Jones whose student ID is 1, and who lives at 1817 Eagleridge Circle in Houston Texas. Her telephone number is 325-098-7654, she is a senior, her date of birth is 07/14/86, and her faculty advisor is Teresa Marx. Tammy has been enrolled at Lipscomb since 08/24/03.

- term.data: has an ID that links terms to different course offerings. The table contains a text description of each term and a status column that shows whether enrollment is open or closed. The first record shows a term ID of 1 for the Fall 2006 term, with enrollment status as closed. This term began on August 28, 2006.

- course.data: shows course records. The first course record, with course ID 1, has the course number IT 101 and is named "Intro. to Info. Tech." It is a 3-credit course.

- course_section.data: shows course offerings for specific terms and includes columns that display the course ID, section number, ID of the instructor teaching the section, and day and time of meeting. The C_SEC_START and C_SEC_END columns store the time each course section starts and ends each day of meeting, respectively. The MAX_ENRL column specifies each course section's maximum allowable enrollment. The first record shows that C_SEC_ID 1 is section 1 of IT 101. It is offered in the Fall 2007 term and is taught by Mark Zhulin. The section meets on Mondays, Wednesdays, and Fridays at 10:00 AM in room CR 101. The class ends at 10:50AM, and has a maximum enrollment of 140 students.

- enrollment.data: shows students who enrolled in each course section and their associated grade if one has been assigned.

QUESTION 1: Defining the Database [40 points]

Define a database schema by constructing SQL CREATE TABLE commands for each of the following relations:

- **student**, which stores information about students. Attributes: S_ID, S_Last, S_First, S_MI, S_Address, S_City, S_State, S_Zip, S_Phone, S_Class, S_DOB, F_ID, Date_Enrolled.
- **faculty**, which stores information about faculty. Attributes: F_ID, F_Last, F_First, F_MI, Loc_ID, F_Phone, F_Rank, F_Super.
- **course**, which stores information for each course offered. Attributes: C_ID, C_No, C_Name, Credits.
- **course_section**, which stores information for each section of a course offered. Attributes: C_Sec_ID, C_ID, Term_ID, Sec_Num, F_ID, C_Sec_Day, C_Sec_Start, C_Sec_End, Loc_ID, Max_Enrl.
- **enrollment**, which stores information about which students are in each course section and their associated grade if one has been assigned. Attributes: S_ID, C_Sec_ID, Grade.
- **location**, which stores information about lecture venues. Attributes: Loc_ID, Bldg_Code, Room, Capacity.
- **term**, which stores information about each school term. Attributes: Term_ID, Term_Desc, Status, Start_Date.

You must design the database with appropriate choices of:

- ***Attribute constraints*** (basic data types and additional constraints, such as NOT NULL, CHECK constraints, or DEFAULT values)
- ***Keys***. Choose appropriate attributes or sets of attributes to be keys, and decide on the primary key.
- ***Referential Integrity Constraints***. Determine all foreign keys, and decide what should be done if the tuple referred to is deleted or modified.

Your solution to Question 1 should include:

1. A list of the primary keys and referential integrity constraints for each relation, along with a very brief justification for your choice of keys and foreign keys.

2. A print out of all your CREATE TABLE commands.

3. A brief justification for your choice of actions on deletes or updates for each foreign key constraint.

4. A very brief justification for your choice of attribute constraints (other than the basic data).

QESTION 2: Populating your Database [10 points]

First, transform the data into comma-separated files. Now that you have your relation schemas defined, you can put the data in the comma-separated text files into the database. If the data in the text file matches the relation directly (which should be true for some of the files), you can import the data from the text files into the database relations.

Although you could (on this little database) convert the text files by hand, this would not be feasible if there was a large amount of data. Instead, you have to use SQL or a graphical user interface (GUI) to do the conversion.

Note that you will need to keep a record of the steps you went through to do the conversion of the comma-separated files. This can just be the sequence of SQL commands you performed, or a user-friendly explanation of the steps taken if you used a GUI.

Note, the data in the text files is all consistent – I believe (or at least hope) that I have removed all errors from it. In a real situation, there are likely to be errors and inconsistencies in the data, which would make the data conversion process a lot more cumbersome.

Your solution to Question 2 should include a description of how you performed all the data conversion, for example, a sequence of the SQL commands that accomplished tasks.

QUESTION 3: Checking your Database [20 points – 2 points each]

You should now check that your database design enforces all the appropriate consistency checks. Each of the following update operations should result in error(s). Perform each of them on your database and record the error message(s) it gives. If you have given names to all your constraints, the error messages will be more informative. For each update, briefly state the constraint(s) it violates. If it doesn't give an error, then your database probably isn't quite right yet. You should at least say what the constraint(s) ought to be, even if you don't get around to implementing it.

a. Insert the following tuples into the **course_section** relation:
   i. (12, 2, 6, 2, 2, "MTWRF", 10:00 AM, 11:30 AM, 5, 35),
   ii. (12, 2, 6, 2, 2, "MTWRF", 9:00 AM, 10:30 AM, 6, 35),
   iii. (2, 1, 4, 2, 3, "TR", 9:30 AM, 10:45 AM, 4, 35).

b. Insert the following tuples into the **faculty** relation:
   i. (4, "Brown", "Colin", "D", 11, "3253456789", "Assistant", 4, 9871),
   ii. (6, "Reeves", "Bob", "S", 15, "3256789012", "Full", , 1234),
   iii. (6, "Reeves", "Bob", "S", 10, "3256789012", "Assistant", 7, 1234),
   iv. (6, "Reeves", "Bob", "S", 10, "3255678901", "Assistant", 2, 1234).

c. Insert the following tuple from the **course** relation: (4, "CS 120", "Intro. to Programming in C++", 3).

d. Delete the following tuple from the **location** relation: (11, "BUS", "222", 1).

e. Delete the following tuple from the **term** relation: (4, "Fall 2007", "CLOSED", "28-AUG-07").

Your solution to Question 3 should include a list of all the SQL updates, the results of the updates, and a brief statement of the kind of constraint that is being violated.

Note: Upon completion of Question 3, you may need to revisit your table definitions (done in Question 1), to include pertinent integrity constraints, before printing your CREATE TABLE commands.

QUESTION 4: Simple Database Queries [33 points – 3 points each]

Your solution should include, for each query, a sequence of SQL commands for the basic queries and the views/tables you created, the output of the final query (for those that are meant to show output), and the SQL message to it from the prompt.

a. Which students (S_ID, S_LAST, S_FIRST) have A's and B's? (Use the IN and/or NOT IN comparison operators.)

b. Retrieve the terms for the 2007 academic year. (Use the LIKE comparison operator.)

c. List the building code, room, and capacity for all the rooms. Sort the result in ascending order by building code then by room.

d. Suppose LIPSCOMB charges $730.00 per credit hour for tuition. To determine how much tuition a student is charged for a class, you can simply multiply the number of credit hours earned for a course by the credit hour tuition rate. For each course, list the course number, course name, and tuition charge.

e. In one query, use group functions to sum the maximum enrollment for all course sections and calculate the average, maximum, and minimum current enrollment for the Summer 2008 term.

f. What is the total number of courses for which student Lisa Johnson has received a grade?

g. Use the GROUP BY clause to list the building code and the total capacity of each building, but only for those buildings whose total capacity exceeds 100.

h. For each student, list the student ID, student last name, student first name, faculty ID, and faculty last name.

i. List the last names of faculty who are teaching in the Summer 2008 term.

j. List all the courses and grades for a student by the name Tammy Jones. Tammy doesn't remember her ID. She also doesn't remember all the courses she took.

k. Create a query that returns the union of the student and faculty tables over the attributes s_last, s_first, and s_phone from Student, and f_last, f_first, and f_phone from Faculty.

QUESTION 5: Slightly Complex Database Queries [40 points – 5 points each]

Your solution should include, for each query, a sequence of SQL commands for the basic queries and the views/tables you created, the output of the final query (for those that are meant to show output), and the SQL message to it from the prompt.

a. Create a nested query to retrieve the first and last names of all students who have the same S_Class value as Jorge Perez.

b. Create a nested query to retrieve the last and first names of all students who have enrolled in the same course sections as Jorge Perez.

c. Create a nested query to retrieve the last and first names of all students who have the same S_Class value as Jorge Perez and who have also been enrolled in a course section with him.

d. A *nested subquery* is a subquery which contains a second subquery that specifies its search expression. Use a nested subquery to create a query to retrieve the names of students who have taken courses with Jorge Perez in the CR building.

e. Create a Union query that displays the names of courses taken by students who were not Seniors, in addition to courses that were offered in Term 6.

f. Use the Intersect set operator to create a query that satisfies both requirements of Question 5(e).

g. Use the Minus set operator to create a query that retrieves the courses that were taken by Freshmen, Sophomores, and Juniors, but were not offered in Term 6.

h. List the names of all junior faculty members and their supervisors. (Think hard on this one!!!)

QUESTION 6: Experimenting with Views [21 points – 3 points each]

Your solution should include, for each query, a sequence of SQL commands for the basic queries and the views/tables you created, the output of the final query (for those that are meant to show output), and the SQL message to it from the prompt.

a. Create a view named **faculty_view** which contains all of the faculty columns except F_PIN.

b. Insert the following tuple into **faculty_view**: (6, "May", "Lisa", "I", 11, "3256789012", "Assistant").

c. Retrieve all the tuples of **faculty_view** to confirm that the new faculty member (May, Lisa) is included.

d. Explain the effect of insert operation in (b) to the database. Why is this so?

e. Create a query that joins **faculty_view** with **location** to list the names of each faculty member, along with the building code and room number of the faculty member's office.

f. Remove **faculty_view** from your user schema.

g. Explain the effect (if any) of (f) to the database.

QUESTION 7: Updating the Database [6 points – 3 points each]

Your solution should include, for each query, a sequence of SQL commands for the basic queries and the views/tables you created, the output of the final query (for those that are meant to show output), and the SQL message to it from the prompt.

a. Change the room to *BUS 211* for all courses taught by *Brown*.

b. Create and fill a new table, called **enrollment_numbers**, which shows each course number and the number of students enrolled in it per section for the Spring of 2008. Display the **enrollment_numbers** table.

How to submit your group projects?

You can do the following 4 submissions - I will grade each submission and then summarize your total score:

1. Solution to Question 1 – Oct. 26.
2. Solution to Questions 2 & 3 – Nov. 2nd.
3. Solution to Questions 4 & 5 – Nov. 9th.
4. Solution to Questions 6 & 7 – Nov. 16.