Fall 2010 CS 3200 Class Project: Milestone 5

The goals for this milestone are (1) to create data to fill our database tables and (2) to practice SQL.

This milestone is to be completed individually (i.e., no teams). You can discuss problems with other students, but you have to create all deliverables yourself from scratch. In particular, it is not allowed to copy somebody else's code or text and modify it.

The report for this milestone is due on Friday, **October 29 at 5pm**. For late submissions you will lose one percentage point per hour after the deadline. This milestone is worth 10% of your overall homework score. Please email the deliverables to both me and Yue. You should receive a confirmation email from either of us. If you have not received a confirmation email within 12 hours after submitting your solution or by the time of the deadline, whichever comes first, you need to email us immediately to make sure we actually received your submission. (Of course, if you submit too close to the deadline, you might receive a confirmation sometime within the next 30-60 minutes after you submitted.) If you need to send multiple files, please create a single zip file. Many other attachments types, in particular rar files, are rejected by the CCIS mail server.

Database Update

For this milestone we will work with the reference database from Milestone 3. Before starting the other parts, **first** make the following modifications in the reference database tables:

1. Add a new attribute exchangeID of type int to table Exchange.
2. Make exchangeID the primary key of Exchange, making sure no other attribute is part of the primary key any more.
3. Delete attributes lender, borrower, ItemID, and DateID from table Rating.
4. Add attribute exchangeID to table Rating and set up a foreign key reference to the same attribute in table Exchange.
5. Make sure (exchangeID, rater, rated) together are the new primary key of table Rating.

Data Generator

Write a data generator to populate the database tables. Before you start, think about how to get the data into your tables. We recommend you write a separate program, e.g., in Java, to create ASCII text files. Then you can import these files into your tables. The SQL Server data import utility can be executed through right-click (on your database) -> Tasks -> Import Data. You can then choose to import from a Flat File Source.

The import utility is quite flexible in accepting various row and column separators for a given ASCII file. Make sure you choose separators that do not occur in data fields. E.g., if you separate columns by comma (,), but have an address field value containing a comma, then the import utility might confuse it for a column separator. Make sure you test the import process first with hand-crafted examples. This way you avoid writing the perfect data generator, just to realize later that it produces the wrong files or file structure.

The goal for the data generator is to create a good amount of interesting and realistic data. Here are a few suggestions and requirements to achieve this:

1. People, rooms, buildings: Your data generator should create meaningful tuples. Create at least 3 dorm buildings and enough rooms for each building to satisfy the application requirements from Milestone 2. For each building, create at least 10 students, exactly one manager, and at least one counselor. Make sure some buildings have more than 20 and some have fewer than 20 students. Create at least 4 janitors and assign them to dorm buildings, so that at least one janitor works in more than one building. When assigning students to dorm rooms, make sure at least one room in each building has more than one student. At least one room in each building should be empty. For non-students, fill some off-campus address strings in, but make sure at least one manager, counselor, and janitor have NULL addresses.
2. Items: Create at least 10 items of each category. Assign items to owners, making sure that each item has exactly one owner. Some people in each group (students, manager, counselors, janitors) should own multiple items, and some people should own none.
3. Create at least 100 item exchanges. Make sure exchanges are consistent. E.g., only the person currently holding on to an item can pass it along to somebody else. The best way to achieve this is to write a small simulation where you keep track of current item possession. Then your program randomly chooses people to borrow an item from another person as time advances in your simulation (e.g., through a counter variable for hours passed).
4. For about 60% of the item exchanges, create both a lender and a borrower rating. For about 15% create only a lender rating, for another 15% only a borrower rating, and for the remaining 10% create no rating. You can achieve this by generating a random number for each item exchange and then determining which ratings to create based on the random number and the required percentage targets. Make sure several numbers of stars are used. For the rating text, insert some random strings consisting of letters and space characters. Make sure some people have average star rating above 3, some between 2 and 3, and some below 2.
5. Create at least 500 messages, most being nice, some not. Make sure most people communicate with at least two other people. Also, there should be several people with zero, one, and more than one friend based on the messages. To ensure this, create enough messages for the last 30 days.

Notice that in your source code you can hard-code certain examples to make sure you satisfy the requirements. But it should be easy to produce larger data sets, if necessary.

For up to 3 extra points, you can also submit a solution for a **bonus assignment**. The extra points can be used to compensate for points lost in the mandatory part of this assignment, but your score cannot exceed 100. The bonus assignment is to also populate the alternative design tables with example tuples. You can try to do this through SQL queries on the “standard” tables.

SQL Queries

After you filled your database with enough interesting data, create the following SQL queries and run them on the database. (If you find it difficult to compute the result with a single query, create temporary tables in your SQL script or use views for intermediate results.):

1. How many students live in each dorm building?
2. What is the ratio of students per counselor for each dorm building with more than 20 students?
3. Which person has borrowed the most books in the history of running the application?
4. For each item (item's ID), who (person's ID) is its owner and who (person's ID) has the item currently?
5. Which person currently holds on to the most borrowed items?

If you are not sure about the semantics of a query, please ask well ahead of the deadline. You can start working on the queries even before the data is generated.

Deliverables

Your report should contain the following items:

1. The updated SQL script for creating all database tables.
2. The source code for generating the example tuples.
3. Screenshots showing each SQL query and its result.

Appendix: Database Remote Access

I talked with our systems support specialists, and they created a nice document describing how to tunnel into CCIS from the outside. You can find it at http://howto.ccs.neu.edu/howto/windows/ssh-port-tunneling-with-putty/. There is a section specifically for SQL Server access. Of course, you will need to use the address of our server there (see Milestone 1).

Once you have set up the tunnel, you should be able to use SQL Server Management Studio from your personal computer. If you do not have it, you can get it from MSDNAA for free. Just make sure it is the 2008 version. Other database clients might also work fine, but we have not tested them. For Linux you will have to use another client, because there is no version of SQL Server Management Studio for Linux. A Web search should help, or you can send an email to the class, asking if anybody else has any suggestions.