CSc 460 — Database Design
Fall 2020 (McCann)

http://u.arizona.edu/~mccann/classes/460

Program #3: JDBC

Due Date: November 5th, 2020, at the beginning of class

Overview: Embedding SQL within another programming language is nice for applications that require more complex calculations or manipulations than plain SQL can handle. Many DBMSes have add–ons that can be used for developing nice windowed applications, but even they may not be flexible enough to create the application you have in mind.

You may have noticed that this is a general election year in the United States. The Center for Public Integrity, and Stateline ("an initiative of the Pew Charitable Trusts"), has collected data from as many states as they could on the polling sites they used in the 2012, 2014, 2016, and 2018 general elections. For this assignment, we will limit ourselves to Montana. Why Montana? Its data set is consistently–formatted, has complete data for all four years, and isn't large (Montana has relatively few polling sites).[1] The data is available on lectura in four `.csv` files; see the Data section, below, for the details. Part of this assignment is to write a program that reads the CSV files and inserts the data into tables within Oracle. The rest is using a combination of SQL and Java to answer a set of queries.

Assignment: For this assignment, you will need to do the following, most likely in this order:

1. Within your Oracle database, create four tables, one for each year, with suitable table and attribute names.
2. Insert the files' content into the relations, being sure to skip the first line of each file (those lines provide the field names).
3. Write a Java program that offers the user a menu of queries that can be asked of the database of polling sites, and uses JDBC to interact with the database to produce the correct answers. The queries are:
   (a) For each county name in the 2012 relation, how many precincts were in that county? (There is no user input for this query.)
   (b) For a given year (that is, a year provided by the user of your program), what are the top five municipalities by numbers of precincts? Display the names of the municipalities and their quantities of precincts in descending order by quantity. Thus, the first one listed will have the most precincts.
   (c) For a given year, how many of the precinct names contain the exact words CHURCH, HALL, SCHOOL, or COLLEGE? For each of those words, in alphabetical order, list the word and the quantity of precinct names that contain it. If a name contains more than one of those words, the name is to counted in the totals for all of the words. For example, if there is a precinct named "BOB'S CHURCH HALL SCHOOL AND COLLEGE, it would be included in all four totals.
   (d) Because we have four years, there are three pairs of subsequent years: 2012 and 2014, 2014 and 2016, and 2016 and 2018. Which counties did not lose precincts in all three pairs of subsequent years, **and** which counties did not gain precincts in all three pairs of subsequent years? For example, imagine that county Armagh had 14 precincts in all four years. We would include Armagh in both lists. Another example: Imagine county Tyrone had 6 precincts in 2012, 5 in 2014, 5 in 2016, and 4 in 2018. Tyrone would be in the list of counties that did not gain precincts in all three pairs of subsequent years. Note that it's possible for a county to be in neither list.

(Continued...)

[1] Why not Arizona? Unfortunately, the data set doesn't currently include Arizona, perhaps because Arizona does a lot of its voting by mail.

Data: I have created local copies of the files for you; you can find them in the `/home/cs460/fall20` directory on lectura. The file names are `mtpollingplaces20NN.csv`, where "NN" is the last two digits of the year (i.e., 12, 14, 16, and 18). Note that the first line of each file has the field names; be sure to skip that line when you read the data.

You **MUST** code your programs/scripts that read the CSV files to read those files directly from that directory! (Why? It makes it easier for the TAs to run your code, and not having dozens of copies of those four files saves a lot of disk space.)

The file formats should be consistent across the years, but there may be some issues you may need to address before you can create relational tables from the data. What issues? There may not be any; there may be several. The TAs and I don't know (or will feign ignorance if we do know!). Finding the issues, if any, and asking us what should be done about them, is part of your task.

Also up to you is the menu format, the structure of the requests for user info needed by the queries, etc. So long as your interface is clear and instructions for the user are complete, we'll be happy. The TAs will grade your program by running it and trying a variety of queries.

Output: Your application is to display the output of a question in a clear, easy to read format of your choice. Remember, you're writing your own program; you are not limited by Oracle's output formatting.

Hand In: You are required to submit a `.tar` file of your well-documented application program file(s) — including any code written to automate the data cleansing process, although that code need not be well–documented — via turnin to the folder `cs460p3`. Name your main application program's `main()` class `Prog3`, so that we don't have to guess which file to compile, but feel free to split up your code over additional files as appropriate for good code modularity.

Want to Learn More?

- An article about the data collection project can be found here:
  https://publicintegrity.org/politics/elections/ballotboxbarriers/data-release-sheds-light-on-past-polling-place-changes
- The source of the Montana polling data files is:
  https://github.com/PublicI/us-polling-places/tree/9f0f3c2abe848170d9f320fd60dc3d7e1631096b/data/Montana

Other Requirements and Hints:

- Because we will be grading your program on lectura using Oracle, it needs to run on lectura and use Oracle. You're welcome to develop your code on your own computer, but be sure it runs correctly on lectura before you submit it for grading.
- If you wish to share any necessary data conversion and "scrubbing" chores with your classmates, that's fine. Stop collaborating when you start coding your querying application. In your documentation, be sure to credit those who helped you with the data organization. Please DO NOT post scripts, etc., on Piazza; we don't want one person doing all of the dirty work for the entire class!
- It is also OK to share query results on Piazza. Doing so can help you discover that your query isn't finding everything it should be finding.
- Make certain that ...
  - ...your database tables are accessible to the TAs (by `GRANT`ing us `SELECT` privileges) to each of your tables: `GRANT SELECT on tablenamegoeshere to PUBLIC;`
  - ...your relations are prefixed with "`yourNetID.`" in your application so that we can execute your program against your database, just as my tables for Homework #3 were accessible with the "`mccann.`" prefix. Failing to do this will cost you points.
- Avoid the temptation to wait to start writing the JDBC code and your application program until you have all of the data loaded into tables. You can (and should!) create and populate small tables for testing purposes early in the development process.