School of Computer Science, McGill University

COMP-421 Database Systems, Winter 2022

Project 3: Application

Due Date Mar 15, 12:00noon

This is the last deliverable for your database application project. If you end up adding more data for this project deliverable, that is OK, as long as you do not EXCEED the record limits imposed in project 2.

Your project 3 work should be a continuation of your project 2 work. You are allowed some model adjustments if that is needed to make your application work correctly. All of this work will be still using DB2 as the database.

In this project deliverable you will be primarily developing a java application to be used by the Midwife (although it will perform only a subset of tasks that a real-world Midwife needs to do).

Please read through the complete description before you start strategizing/working on individual problems.

The Assignment

Some of the solutions to the below questions have to be provided in a document **project3.pdf** whereas others would be independent files of their own (described next to each question).

1. Make sure you are using the DB2 database provided in the winter2022-comp421 server for your project (we have mechanisms to verify this). Not doing this and using another database (including PostgreSQL), using databases installed in your own personal computer, etc., will result in **5 points penalty** (even if you had this penalty in the previous deliverable).

2. Include the relational model that you are using for this phase of the project - even if you have not made any changes from project 1 and 2. Include this in the document **project3.pdf** under the section **Relational Model**. You need not include any restriction/assumption information along with this. Not doing this will result in **1 point penalty**!

3. **(21 Points)** write a java program `GoBabbyApp.java` - you can base this on the example JDBC program that is given in mycourses as part of the JDBC tutorial. You can add other java/class files, etc., but this is your *main*. This a simple console (command line) based program with a menu through which you can "loop". User selects the menu option based on what they would like to do, program performs some action(s) and goes back to displaying a menu. You are free to format/phrase the menu options in sensible ways that do not deviate from the intent of the menu or increase the burden on the user of the application.

   When the application starts up, it will ask the midwife to enter their "practitioner id"

   ```
   Please enter your practitioner id [E] to exit:
   ```

   If the user types **E**, the application will exit. (Make sure it closes any active database connections, etc., before terminating.)

   If the practitioner id entered is not valid (i.e., does not exist in the database), the application will throw an error message (up to you) and return back to the same menu.

   If the practitioner id is valid, it will prompt the midwife to enter a date for the appointments she wants to see.

   ```
   Please enter the date for appointment list [E] to exit:
   ```

   At this point, your application should list all the appointments for that date for that midwife, ordered by time. An additional column whose value is P or B is also to be displayed to indicate whether she is the primary or backup midwife for that pregnancy (for which the appointment is for). If there are no appointments for that date, just print an appropriate message and go back to asking for the date.

An example format is given below:

```
1:  10:00 P Scarlett Walker WALS-5323-5534
2:  13:30 B Jasmine Ali ALIJ-6234-7423
3:  15:00 P Cora Pinto PINTC-7342-7233

Enter the appointment number that you would like to work on.
                  [E] to exit [D] to go back to another date :
```

Here, the output listing should contain only the name of the mother (because you really cannot have a pregnancy without a mother and as discussed in project 1, we are treating a couple as "one working unit") and her health insurance number. How you format time, name, insurance numbers, etc., is left to you and their values/data type/format will be dependent on what you used in Project 2 - the above format is only to show you the intended concept.

Note how we are not showing the "pregnancy number", "couple id", or something of that sorts. It is true that a couple could have multiple pregnancies recorded in the system and that each appointment is associated with a specific pregnancy. However, this is something that your application will have to "remember" when it reads that information from the database (because it may be needed to attach a note, test, etc., depending on your model), but not necessarily something we need to display to the user. I.e., the "pregnancy number", etc., and similar concepts would be used only internally by the application and is not shown to the user as part of the displayed output of this menu.

If the user types **E**, the application will exit. (Make sure it closes any active database connections, etc., before terminating.)

If the user types **D**, the application will go back to the menu where it asks for the date. If the user selects a number, the application would give the following menu.

```
For Jasmine Ali ALIJ-6234-7423   mother name and hCardID

1. Review notes
2. Review tests
3. Add a note
4. Prescribe a test
5. Go back to the appointments.

Enter your choice:
```

Functionality Specs

- **Review notes**
  check if string legnth > 50 then substring, else just print
  If the user chooses **1**, the application will list all the notes relevant for this pregnancy in the descending order of date time, the output should be date-time and no more than the first 50 characters of each note(s). An example format shown below for conceptual purposes.

  ```
  2022-02-13 10:02:45  Baby has good movements
  2022-01-10 14:22:05  Couple would prefer home birth.
  ```

  And then display the previous menu with the 5 options.

- **Review tests**
  If the user chooses **2**, the application will list all the tests relevant for this pregnancy (but only the tests relevant for the mother - this is a simplification) in the descending order of date (test prescription), the output should be date (test prescription), test type and no more than the first 50 characters of each result(s). An example format shown below for conceptual purposes (put square brackets or something like that in your output display to separate from the type of test and the text for results). If a result is not yet available, display the text `PENDING` in it's place.
  if NULL, make it pending?

  ```
  2022-02-01 [blood iron] A bit low, recommended supplements.
  2022-01-20 [first trimester ultrasound] Baby normal growth.
  ```

And then display the previous menu with the 5 options.

- **Add a note**
  If the user chooses **3** the application will let the user type in a text (note) that is then stored into the system.

```
Please type your observation:
```

  After which it will display the previous menu with the 5 options.

- **Prescribe a test**
  If the user chooses **4**, the application will let the user enter the type of test that is being prescribed and that will be recorded in the system. Again, for simplicity, we will assume that test is for the mother.

```
Please enter the type of test:
```

  For simplicity, you can assume that the prescription date and sample date of the test is the date on which the test prescription is being entered.

  After which it will display the previous menu with the 5 options.

- **Go back to the appointments**
  Goes back to the previous menu and displays all the appointments once again for the date that was originally entered by the user.

Design Details

- When implementing the "details" of each functionality, you are free to follow a programming approach that intuitively works with the intent of each functionality in how the user is asked to enter any additional information. However, the application should not probe the user for any information that it can already compute from the data stored in the database or is supposed to generate by itself. For example, the use should not have to enter "test id", "note id", etc. (as this is automatically generated by the system). Another example, The system should also not ask the user to enter information like "pregnancy - id" or "couple id", etc., when entering a note/test, as that information should have been already "remembered" by the application when it navigates through the various menu options.
- You need not have to handle errors related to data formatting (i.e. user entered date where number was expected, etc.) or worry about a user deliberately trying to break your system. You can assume that once a user selects an option they will provide all the necessary information required for the application to act up on that option if the application prompts the user to do so.
- You need not produce any messages for successful execution of actions but is free to do so, it might help debugging your application easier. But keep it short and ensure that it is not adding clutter in the screenshots (or turn it off once you are done developing and fixing your code).
- No errors should result in the application crashing and terminating.
- Please keep in mind that for a given option you might have to write data to multiple tables or read from multiple tables or may need a combination of reads and writes. You need to handle this in your application according to your model and schema implementation.
- You are graded primarily by the functionality and not how "pretty" the application looks. Do not spent time on cosmetics before you get the functionality under control (but it should be "readable" and not a mess).
- It is assumed that any additional information/data that you need to support the application (eg. couple, midwife information, etc.) are already present in the database. They need not be added/maintained through your java application. You can insert them separately (reuse the template scripts from project 2 to make it easy).
- You are not obliged to do any "checks" and "error handling" besides the ones required for the successful execution of the application flow that is given below.

Execution Flow of your Application, Screenshots, and things to turn in

While you may do the development and testing of your java program in any manner you want, **for the purpose of submission screenshots, it MUST be executed from the command line of the Unix server (not IDE) used for comp 421**. **Not doing this will incur a 5 point penalty!**

Where it says "run a query in the database", **you should strictly use the db2 command line to run your queries.** Not doing this and using IDE will incur a **5 point penalty**. The query and its corresponding output should be visible in the SAME screenshot - otherwise it would receive 0 points. The screen shots below should be placed under a section **Application interaction** in **project3.pdf**. Make sure to label each screen shots with the corresponding question numbers so that TA can easily identify which screenshot is associated with each question.

In the below description, replace place holder variables such as $\mathbb{P}$, etc., with a sensible data value depending on your data model. Merely leaving them as $\mathbb{P}$, etc., will result in point deduction.

In each question below (`a`, `b`, etc.,), each step (`i`, `ii`, etc.,) must be answered by at least one (but possibly many) screenshot. Make sure to label the question and step numbers when you paste the screen shots into **project3.pdf**. The screenshots of application should also include its response to your menu choice/input after it has received and processed it and not merely you typing an option/information at the application prompt.

This discussion is for steps where it says "Run a query in the database". You must execute them using one single SQL query that produces all the required information (and not multiple queries). You might find outer joins useful (depends on your model) for many such queries below. Make sure that your queries are not merely dumping all the records but only those relevant to the question being asked. For example, if question requires you to verify something about the individuals $\mathbb{P}_1$ and $\mathbb{P}_2$, the query output should not contain information about others ($\mathbb{P}_3$, etc). You only need to include relevant columns - required to uniquely identify entities and relationships, and any new/modified information as performed by your application in the previous step(s) in the query outputs. You may truncate the length of large text columns (using `SUBSTRING`, etc.) if it reduces the clutter, as long as the required information is easily identifiable.

(a) **(2 points)**

i. Run a query in the database that shows that midwife with practitioner id $\mathbb{P}_0$ does not exist.

ii. Start the application and enter the practitioner id (of the midwife) $\mathbb{P}_0$.

(b) **(12 points)**

i. Run a query in the database that shows that a mother $\mathbb{M}_1$ has had at least two pregnancies (can be as part of the same or different couple - does not matter).

ii. In the application, enter the practitioner id (of the midwife) $\mathbb{P}_1$ (must exist in the database).

iii. Enter the date $\mathbb{D}_1$. One of the appointments must be for $\mathbb{M}_1$ and another one for $\mathbb{M}_2$, a different expecting mother.

iv. Choose the appointment number associated with $\mathbb{M}_1$.

v. Enter **1**. (Should show at least two notes).

vi. Enter **2**. (Should show at least two tests).

vii. Enter **3**. Add a note $\mathbb{N}_1$.

viii. Run a query in the database that shows all the notes associated with $\mathbb{M}_1$, include couple information, pregnancy number, etc., as relevant in your model. The actual note text may be truncated to first 50 characters or so to reduce clutter.

ix. Enter **1**.

x. Enter **4**. Add a test for *trisomy 21*.

xi. Run a query in the database that shows all the tests associated with $\mathbb{M}_1$, include couple information, pregnancy number, etc., as relevant in your model. The result attribute of the test maybe truncated to the first 50 characters or so.

xii. Enter **5**

(c) **(5 points)**

i. Run a query in the database that shows all the tests associated with $\mathbb{M}_2$, include couple information, pregnancy number, etc., as relevant in your model. The result attribute of the test maybe truncated to the first 50 characters or so.

ii. Choose the appointment associated with mother $\mathbb{M}_2$

iii. Enter **1** (to see the relevant notes).

iv. Enter **4**. Add a test prescription for *Group B streptococcus* (test type).

v. Run a query in the database that shows all the tests associated with $\mathbb{M}_2$, include couple information, pregnancy number, etc., as relevant in your model. The result attribute of the test maybe truncated to the first 50 characters or so.

(d) **(2 points)**

i. Choose **5**, followed by **D**.

ii. Enter a different date $\mathbb{D}_2$.

iii. It should show at least two appointments for mothers $\mathbb{M}_3$, $\mathbb{M}_4$, who are not the same as previous examples.

Your screenshots should be clear at 100% magnification of the document, without requiring to zoom in. The text for queries, outputs, and program options must be easily readable. Otherwise they will be treated as not submitted.

4. **(2 Points)** This is not part of your java program. Create an index that will help retrieve a mother's address, assuming that we only have a phone number to start the search with. Turn in a screenshot of the index creation (using db2 command line) under a section **Indexing** in your **project3.pdf**.

5. **(2 Points)** This is not part of your java program. Write a query that will produce the number of births for each month (only count babies born). Produce a chart with month (could be the numeric or name) on X axis and count on Y axis. Remember to label your axis. Should have at least three months worth of data and each month should have more than 5 births - ideally, different number of births per month. (You may go up to 100 records for associated tables, if needed). Include the query that you used and the image of the chart in your **project3.pdf** document under a section **Data Analytics**. For this question, you may use spreadsheet programs like excel, google docs, or use packages like matlab, matplotlib, etc., or specialized analytics softwares such as `Tableau`. Also turn in your " excel workbook" or "program script", etc., as a separate submission file. Remember to include the name of this attachment in the project document.

   You will find the links to two short tutorials in mycourses (under `Database` → `Simple Data Visualization` folder) on how to create pivot charts in Excel and Google spread sheets.

   You can export the required data from DB2 tables using the db2 command line utility (**not from IDE**) using its `EXPORT TO` syntax. You may also export the data in any other way that is convenient to you (but the data must be extracted using a SQL query and from the database).

Files to Submit

Please use this as a check list to ensure that you do not miss uploading any required files!

Your submission will contain the following files:

1. **GoBabbyApp.java** This is the java program that you developed. If you had additional java files, include them as well. Missing any file from your java program will result in a 5 point penalty!.

2. **project3.pdf** this will contain your current relational model, any screen shots, discussions, etc., as indicated under various questions. Make sure you put them under the correct section headings so that TAs can match your solutions to the questions. Haphazard clutter in the document can result in point deductions.

3. The **workbook** (Excel, etc..) or program script that you used to generate the chart. Missing this will result in 0 points for your question 5.

These are the only file formats that you are allowed to use for submission! All your submissions must be present in MyCourses. Links to websites, cloud drives, etc., are NOT acceptable! Any such documents must be downloaded and submitted to the submission folder by the student.

You may submit these files separately or tar them. Resubmissions are allowed (please do not email me saying you do not know how to resubmit files). Please try to submit an entire set, so that TAs do not have to find your files across multiple submissions.

You do not have to submit your new data / schema SQL files even if you added new data, modified the table definition, etc. for project 3.

Please turn in your submission in mycourses under project 3. Submissions to wrong folders may not get graded. **You are responsible to download and verify that your submission is correct (not corrupted or incorrect file, etc).** There will be no accommodations if you decided to ignore doing due diligence (you will get a 0). Project grades also influences the pass/fail criteria of the course. Please review the course outline for any of these details.

Important Database Etiquette

Please remember that the server `winter2022-comp421.cs.mcgill.ca` and the databases installed them are meant to be used for the course work of COMP 421. Using these systems for other work (including other course work) is not allowed. You are sharing these resources with other classmates, so be mindful of its proper use.

Please go through the `README` file in MyCourses under "Database" → "Connecting to DB Servers" for comprehensive list of restrictions. Not following them may result in your id being disabled as stated in the course outline.

Questions ?

Please use **Ed** for any clarifications you need (`projects` → `p3`). You may also want to review the project 1 requirements to refresh some of the concepts and terminologies. Do not email the instructor or TAs as this leads to a lot of duplicate questions and responses (not an efficient system). Such emails will not receive any replies.

The only emails you are allowed to write is to your grading TA if you have questions on the feedback you received (as they are personal in nature) - do not post them in Ed, your grading TA may not see it.

Please check the pinned post "P3 general clarifications" in Ed before you post a new question. It might have been already addressed there, in which case we will not address it again.

Questions about general clarifications must be marked public (as other students will also benefit from this and may even have a valid response). TAs and Instructors upon their discretion may toggle any private posts into public mode for the benefit of the student population at large.

There will be specific office hours for the project that will be announced closer to the due date.

Extensions and Late submissions

- Remember, your submission is **due on Mar 15, 12:00 noon**. There is no place for excuses.
- A maximum of 2 days of late submission is allowed with a penalty of 20% of the achieved grade per day (rounded up, even for a minute).
- Penalty waivers are granted only for medically documented emergencies and under any circumstances **will not be granted unless requested 24 hours before the due**. I expect you to be better organized and get the work done ahead and leave the last 24 hours only for one last final check.
- There is no "partial late penalty" concept, it is applied to your entire submission, and not just specific questions that you submitted late.