School of Computer Science, McGill University

COMP-421/764 Database Systems, Winter 2021

Project 3: Application

Due Date Mar 16, 05:00pm

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.

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. (0 Points) 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.

2. **(20 Points)** write a java program `VaccineApp.java` - you can base this on the example JDBC program that is given in mycourses as part of the JDBC tutorial. 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 that action and goes back to displaying the menu.

   Below is an example of what this menu would look like, but you are free to format/phrase it in sensible ways that do not deviate from the intent of the menu.

```
VaccineApp Main Menu
        1. Add a Person
        2. Assign a slot to a Person
        3. Enter Vaccination information
        4. Exit Application
Please Enter Your Option:
```

   Functionality Specs

   - **Add a Person**
     This is the instance where a (new) person is registering with the system. When this option is selected, your application should allow the user to enter (type) the relevant information. If the user's health insurance number is already associated with someone else, the application should stop adding the new person and instead respond with an appropriate message to the user. At this point the system should prompt the user whether they wish to update the existing Person information associated with that insurance number with the newly entered details. If the user chooses to do so, then the existing information is updated. Otherwise, no action is taken.
   - **Assign a slot to Person**
     This is used when a Person has to be assigned a slot (whether it is their first dose or later doses). Prompt the user for any necessary input and update the slot allocation information in the database. The application should not allow the user to assign a slot to this Person, if the said Person has already taken the required number of shots for that particular brand of vaccine (if it is not their first shot). You can assume that a person is never vaccinated by two different brands. Do not assign a slot if that is already given to someone else or is in the "past" (for simplicity, you may interpret this as a slot from a day that

is before "current date", if you do not want to get into the time level granularity - but you are free to do that.)

- **Enter Vaccination information**
  This is where we "record" in the system that a certain Person was vaccinated. Again, you need to record appropriate information for this (i.e., the nurse, vial info, slot, etc. - complete details depends on your model). Stop the entry from being made if the vial is from a vaccine brand that is different from previously applied vaccine brand for that Person (if the Person had received a vaccine shot before).
- **Exit Application**
  The application terminates. Make sure it closes any active database connections, etc., before terminating.

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. For example, the application should not ask the user to enter the number of previously applied doses for a Person or the brand of vaccine that was used. This can be easily retrieved from the stored data once you received a Person's basic information from the user (such as the health insurance number).
- 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.
- In any instance where the system "refuses" to do a task (as in for example the insurance number is already assigned to someone else) it should give a proper message to the user. You need not produce any messages for successful execution of actions but is free to do so, it might help debugging your application easier.
- Make sure the application loops back to the main menu once it is done taking care of an action request. 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.
- It is assumed that any additional information/data that you need to support the application (eg. vaccine batches, hospital info, nurses, 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 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

You may use the db2 command line or an IDE to run your queries, as long as a query and its corresponding output are visible in the SAME screenshot. 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 $\mathbb{H}$, $\mathbb{P}$, etc., with a sensible data value depending on your data model. Merely leaving them as $\mathbb{H}$, $\mathbb{P}$, 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**.

This discussion is for steps where it says "Run a query". 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 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.

(a) **(5 Points)**

i. Run a query in the database that demonstrates that there is no person with the health insurance number $\mathbb{H}$.

ii. From your application, add a new person $\mathbb{P}_1$, whose health insurance number is $\mathbb{H}$.

iii. Run a query in the database that shows the new information that got added.

iv. From your application, add a new person information $\mathbb{P}_2$, whose health insurance number is $\mathbb{H}$. Choose *Yes* when the application prompts whether the current person's information must be updated. $\mathbb{P}_2$ could be just a slight variation of $\mathbb{P}_1$, such as a different phone number or so (so maybe in reality they are the same individual, and the user is just updating the personal information).

v. Run a query in the database that shows that $\mathbb{H}$ is now associated with $\mathbb{P}_2$'s information.

(b) **(5 Points)**

i. Run a query in the database that shows that a person $\mathbb{P}_3$ exists, but has not received any vaccination shots.

ii. From the application, assign a slot $\mathbb{S}_1$ (this maybe a combination of date, time, location, etc., depending on your model) to $\mathbb{P}_3$.

iii. Run a query in the database that shows that $\mathbb{P}_3$ is now assigned the slot $\mathbb{S}_1$.

iv. Now, from the application try to assign the slot $\mathbb{S}_1$ to another person $\mathbb{P}_4$.

v. Run a query in the database that shows that $\mathbb{P}_3$ is still assigned the slot $\mathbb{S}_1$ and no slot is assigned to $\mathbb{P}_4$.

(c) **(4 Points)**

i. Run a query in the database that shows that the vaccine brand $\mathbb{V}_1$ has 2 recommended doses.

ii. Run a query in the database that shows that a person $\mathbb{P}_5$ has received 2 doses of the vaccine brand $\mathbb{V}_1$ already.

iii. From the application, try to assign a valid free slot $\mathbb{S}_2$ to $\mathbb{P}_5$.

iv. Run a query in the database that shows that slot $\mathbb{S}_2$ is still free.

(d) **(3 Points)**

i. From the application menu, record that $\mathbb{P}_2$ (the person from 2a) is being administered the vaccine $\mathbb{V}_1$, by the nurse $\mathbb{N}_1$ using the vial $\mathbb{W}_1$.

ii. Run a query in the database that shows this information, along with the slot in which this was applied. The output must contain information useful to identify the person, nurse, vial and the slot at a minimum.

(e) **(3 Points)**

i. Run a query in the database that shows that a person $\mathbb{P}_6$ has received 1 dose of the vaccine brand $\mathbb{V}_1$ already.

ii. From the application, try to assign a valid free slot $\mathbb{S}_3$ to $\mathbb{P}_6$.

iii. From the application menu, try to record that $\mathbb{P}_6$ is being administered the vaccine $\mathbb{V}_2$, by the nurse $\mathbb{N}_2$ using the vial $\mathbb{W}_2$.

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

3. **(2 Points)** Create an index that will help retrieve a person's (here person is defined as someone who registers for vaccination) address, assuming that we only have a phone number to start the search with. Will this index be any more practical/efficient than searching by the insurance number of the person (assuming real-world data and millions of individuals) to retrieve the same information? - give a short discussion (2-3 lines). Turn in a screenshot of the index creation (using db2 command line) and the discussion under a section "Indexing" in your **project3.pdf**.

4. **(3 Points)** This is not part of your java program. Write a query that will produce the number of persons vaccinated each day (count only if it is their first dose of vaccinations). Produce a chart with date on X axis and count on Y axis. Should have at least five days worth of vaccination data and each day should have more than 5 vaccinations - ideally, different number of vaccinations per day. (You may go up to 200 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 "workbook" or "program script" 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

Your submission will contain the following files:

1. **VaccineApp.java** This is the java program that you developed.
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 or program script that you used to generate the chart.

All your work should be contained within the submissions made in mycourses. Links to websites, cloud drives, etc., are not permitted.

Please turn in your submission in mycourses under project 3. Only a maximum of 3 late days allowed (15% of the maximum allocated points deducted per day, rounded up). Project grades also influences the pass/fail criteria of the course. Please review the course outline for any of these details.

Questions ?

Please use Piazza for any clarifications you need. Do not email the instructor or TAs as this leads to a lot of duplicate questions and responses (not an efficient system). Please check the pinned post "P3 general clarifications" before you post a new question. It might have been already addressed there, in which case we will not address it again.

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