CS-A1153: Databases

Final Project Deliverable Package (Report)

Group 08

Yanghang Zeng, Licheng Liu

Jiamin Yan, Yue Wan

June 11th, 2022

Table of Contents in the Project

1. Introduction

In this course, we are asked to finish the course project. Our project will research on the distribution of coronavirus vaccination in Finland. The database contains much information, including the batch ID of information, the transportation of different vaccine batches, the vaccinated patients, as well as the staff in medical institutions.

In the first part of the project, we have developed our UML diagram of the database, as well as the relational schema. Then in the second part of the project, we updated our UML diagram, along with the relational schema. Also, there are several questions related to the data processing with the pandas in Python, as well as the SQL queries in both Task 2 and Task 3. Additionally, during the entire process, we have adopted the PostgreSQL to deal with the operations to the database.

The following contents will mention our design choices, performance analysis and how our group members work during the past weeks.

2. Design and Use Cases

2.1 Purpose

The database is based on the Excel form with several sheets, which include the information of vaccine distribution from several health institutions in Finland. This Excel form is already available in GitLab, and what we should do is to create the table with SQL and Python. Once the database is successfully built, we could use data analysis tool in pandas and query codes in SQL. The main purpose of creating this database is to help us analyze the issues related to the vaccine distribution.

2.2 UML Diagram and relational schemas

In terms of the UML diagram, we have finally specified the version in part 2. And the following figures presented our UML diagram for this project, and the relational schemas of our database.

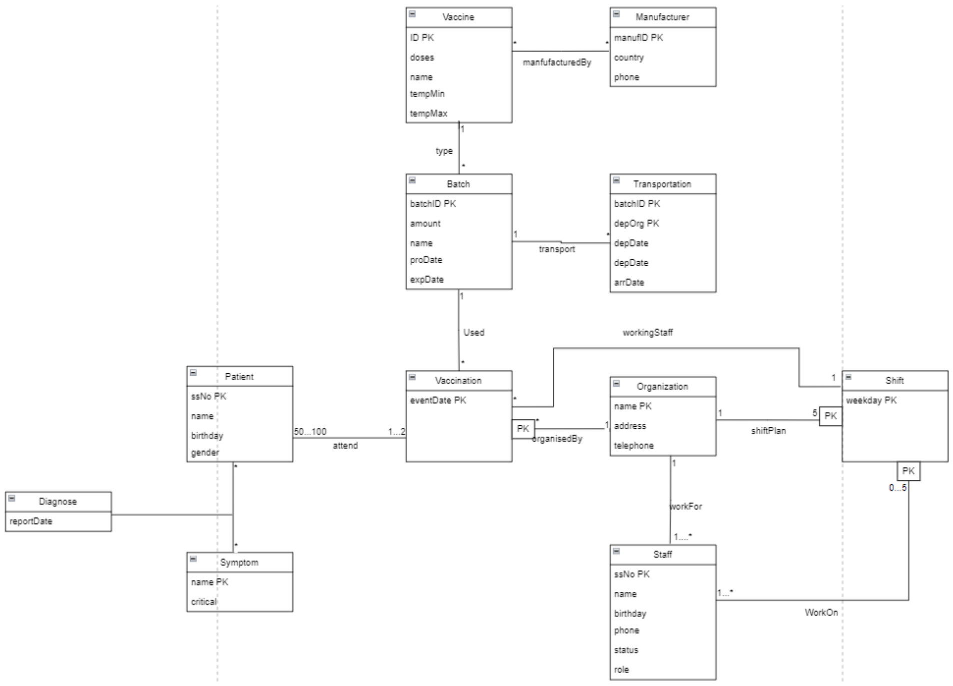
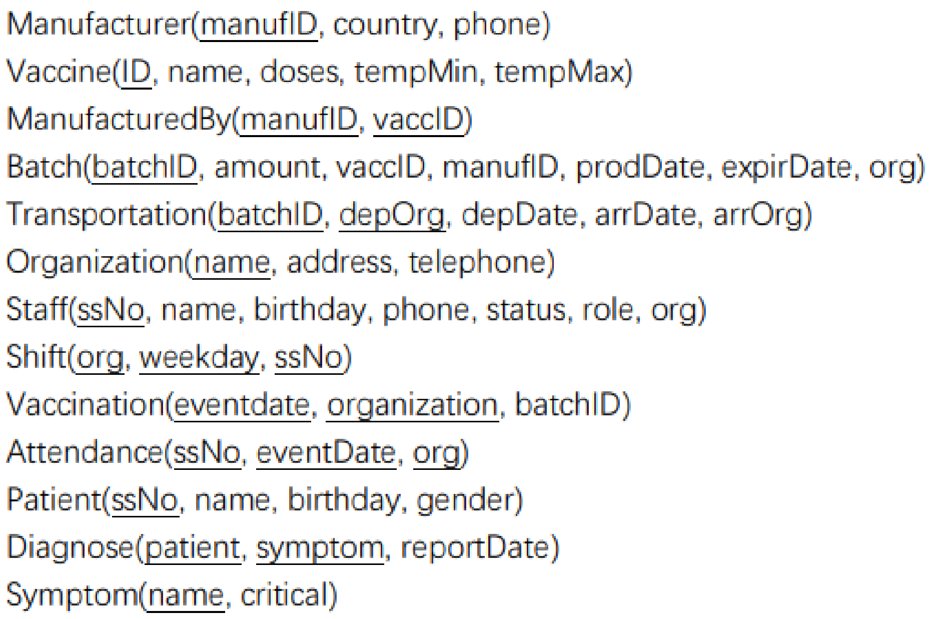


Figure 1. UML diagram



Manufacturer(manufID, country, phone)
Vaccine(ID, name, doses, tempMin, tempMax)
ManufacturedBy(manufID, vaccID)
Batch(batchID, amount, vaccID, manufID, prodDate, expirDate, org)
Transportation(batchID, depOrg, depDate, arrDate, arrOrg)
Organization(name, address, telephone)
Staff(ssNo, name, birthday, phone, status, role, org)
Shift(org, weekday, ssNo)
Vaccination(eventdate, organization, batchID)
Attendance(ssNo, eventDate, org)
Patient(ssNo, name, birthday, gender)
Diagnose(patient, symptom, reportDate)
Symptom(name, critical)

Figure 2. Relational schemas

2.3 SQL File

We wrote a SQL file which was used to create the database, and the name of this SQL file is 'part2_create.sql'. Then in the Python file 'part2_create', after connecting the Postgre database from our computers, which are the local machines, the SQL file is read and the database is created.

As for the another SQL file ‘part2_query.sql’ which is used for the query task in Part 2 of this project, we also firstly wrote SQL codes in this file. Then we will use python to read this file, and the cursor based on the connection to the Postgre database is adopted to operate with the database.

As for the third part of this project, we did some data analysis jobs using the ‘part3_analysis.ipynb’ file, all the results are visible in the file.

Workflow of our code:

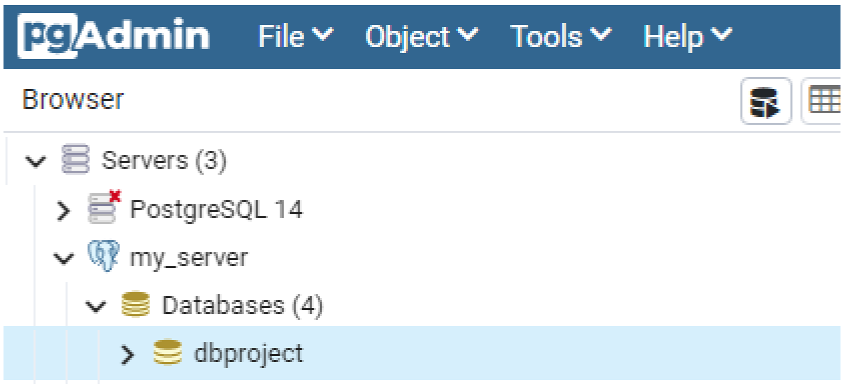
1. Run ‘part2_create.py’ to create tables in remote server and populate the tables with the excel data.
2. Run ‘part2_query.py’ to get results of required queries
3. Run ‘part3_query.py’ to get analysis of the data.

2.4 Creating database(PostgreSQL)

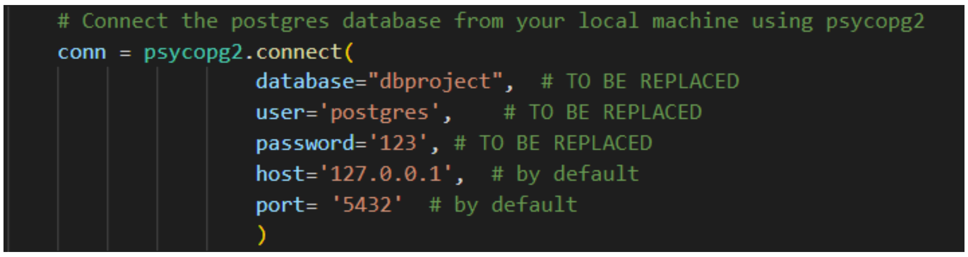
2.4.1 Database on local machine

1. create a server and a database inside it in the GUI tool PGAdmin for PostgreSQL.

1. create a server and a database inside it in the GUI tool PGAdmin for PostgreSQL.



2. run our python program *part2_create.py* to create tables inside the database, and importing data from the excel file to our database. The configuration for building the connection is set in last step, just replace the configuration in the python program, as shown below.



```
# Connect the postgres database from your local machine using psycopg2
conn = psycopg2.connect(
                database="dbproject",  # TO BE REPLACED
                user='postgres',     # TO BE REPLACED
                password='123', # TO BE REPLACED
                host='127.0.0.1',  # by default
                port= '5432'  # by default
                )
```

3. run our python program *part2_query.py*, to get the result of 7 required queries.

2.4.2 Database on remote server

1. replace the configuration part in the python program with the config information of the course server.
2. run *part2_create.py* to create and populate the tables
3. run *part2_query.py* to get the results of 7 required queries

2.5 Assumptions

The basic assumption for this project is that different tables in the database would share the identical attributes, although the names of some attributes in different tables might differ from each other.

For Manufacturer and Vaccine, we assume that a manufacturer can produce arbitrary number of vaccine types, and a type of vaccine can be produced by arbitrary number of different manufacturers.

For the Transportation, we assume that a batch can only be transported once within a day, so we set the batchID and departure date as primary keys. A batch can be transported many times, or wasn't transported since it arrived in the first organization.

For Patient and Diagnose, we assume that a patient can report an arbitrary number of symptoms each time.

For Shift, it will have three primary keys, because there could be many staff working on a certain weekday in a certain organization. And a staff might works for 0~5 days per week.

For organization, we assume that an organization must have more than 1 staff.

2.6 Discussion on design choices

In terms of the design of the database, we have added a relation called "Attendance", which indicates the patients took the vaccine shot in a specific date. We removed the 'status' in the Patient, because it can be queried from other tables to know the number of doses a patient was given.

Secondly, in the relation "Vaccination", the new attribute "organization" as one of the primary keys is added, as well as the new attribute "batchID", so the batch ID could be also available. And this table helps to check the people who comes to take vaccines.

Additionally, in the relation "Shift", we have added several new attributes, including "org", "weekday" and "org"; in the relation "Symptom", we have removed the attribute "reportDate".

The reasons for designing databases for our case are based on the ideas to reduce the data redundancy and improve the efficiency, since some of the attributes appear multiple times in different relations. Also, the modification of original database also considers the need of querying in a more effective way.

3. Performance Analysis and Improvements

3.1 Performance Analysis

This database designed by our group worked well when it's used for data analysis. But there are stills some improvements can be done.

Firstly, our project works on a basis of a small dataset, the data processing program and queries cost a small amount of time. However, the scale of dataset can be massive when the database is used in real cases. As the dataset increases, the query to the database might cost unneglible amout of time. For example, a simple 'SELECT … FROM' query must scan all rows of a table to look for rows that match the given condition. The table could consist of millions of rows or even more, which might take several seconds to scan the whole table.

Secondly, our database doesn't include any transaction to protect our data from wrong state after a modification failure. When a group of operations fails in the middle of execution, its effect will still remain in the database, but the overall effect is not correct. Another case is when multiple operations overlap in time, they might have wrong effect on the database.

3.2 Improvements

To improve the performance of our database when it comes to larger dataset, it is necessary to create some indices on the tables. Creating indices helps speed up the data retrieval operations. What needs to be taken into account is the selection of index, because index itself costs some storage space. It is not possible to create index for each possible cases. Instead, it requires careful consideration and calculation to find out some of the most frequent queries, so that the overhead will be effectively reduced after creating indices.

To prevent unexpected incorrect state in our database, it is necessary to create transactions. For example, grouping operations into transactions ensure the atomicity of all operations. In addition, transactions will also eliminate the overlap of operations, as SQL requires trancastions to be executed in a serializable manner as a default. Moreover, it is useful to set the isolation level carefully to prevent some issues caused by reading and writing.

4. Team cooperation

4.1 Division of our project

Name	Role
Yanghang Zeng	Leader of group; Coordinate the group with all the issues; Contribute to the design of relation schemes and UML diagram, the connection to the Postgre database and complete the code for Part 2, Part 3 and the deliverable package.
Jiamin Yan	Contribute to designing the UML diagram and tasks in the Part 2, Part 3 and the deliverable package
Licheng Liu	Contribute to designing the UML diagram and the presentation of our group project.
Yue Wan	Contribute to solving the tasks in Part 2 and Part 3, as well as design slides for the presentation.

4.2 Estimated schedule of the project

Period	Schedule
April 30th- May 5th	Discussion on the UML Diagram and relational schemas
May 6th- May 8th	Finalize the UML Diagram and relational schemas, as well as other issues in Part 1
May 15th- May 20th	Update the UML Diagram and relational schemas
May 21st- May 24th	Solve the SQL tasks in Part 2
May 31st- June 5th	Solve all the tasks in Part 3
June 8th- June 11th	Finish the final deliverable package

4.3 The way to work as a group

Before we implement tasks in different parts of the project, we would hold a meeting on Zoom, and we will talk the difficult and important points of the project in different stages. Then we will assign tasks to each group member.

During the project, we have used our own group chat to discuss general matters, including the schedule and the coming deadlines of the project. Also, this course provides a great platform GitLab, in which we will push our latest codes to our repository for other members reference. The use of GitLab indeed improves the efficiency of managing the project as a team.

5. Conclusion

In this project, we designed a database for the distribution of vaccine and vaccination data, which includes the relational schema, the program to create the database from data file, and some data analysis tasks.

The advantage of our database is its resonable design of relation schemas which lead to low redundancy. The disadvantages of it are the performance of execution and lack of prevention from error state. Some possible future work on it includes creating indices and creating transactions.

During this project, our group learned about how to design a practical database from scratch and how to solve problems during the coding process. We become more familiar to combining PostgreSQL and python to do data analysis tasks through this project.