Project report

Group 17

Table of Contents

Introduction

This report explains our work on the Databases course project. We set out to design, implement, and test a database system for managing a vaccine distribution network. This project report covers 5 sections: Introduction, Design and use case, Performance analysis and improvements, Working as a team, and Conclusion.

First, we introduce the task which the our database aims to address and describe the functionality we were tasked with implementing. We then go over the database schema and details of the implementation. We then move on to analyse the strengths of the database, present the improvements we made along the development process, as well as potential avenues of future development. Finally, we discuss aspects of teamwork within our group, and conclude with some of the takeaways from the project as a whole.

PostgreSQL relational database management and Python language are used to create tables, manipulate the data, interact with the database, and perform data analysis.

Design and use cases

Purpose of the database and use case

The purpose of designing the database is to provide an organized and efficient structure for storing, managing, and retrieving data. Databases are designed to handle large volumes of structured or unstructured data in a consistent and secure manner. They provide a centralized repository for storing data that can be accessed by multiple users or applications concurrently.

In our project, the purpose is to efficiently store data such as covid-19 vaccine, vaccine manufactures, transporting of vaccines in Finnish hospitals and clinics, vaccination events in Finland, patients and their symptoms, the staff of each hospital/clinic in Finland and their working schedules.

There are various use cases, including data storage and retrieval, data management, data sharing and collaboration, and data analytics. By storing data in database in a structured format, we enable efficient searching, filtering, and data sorting. We can also manage data such as adding, modifying, deleting while keeping data integrity and consistency. The use of database also facilitates data sharing and collaboration among multiple users or systems.

UML diagram and relational schema

First UML diagram

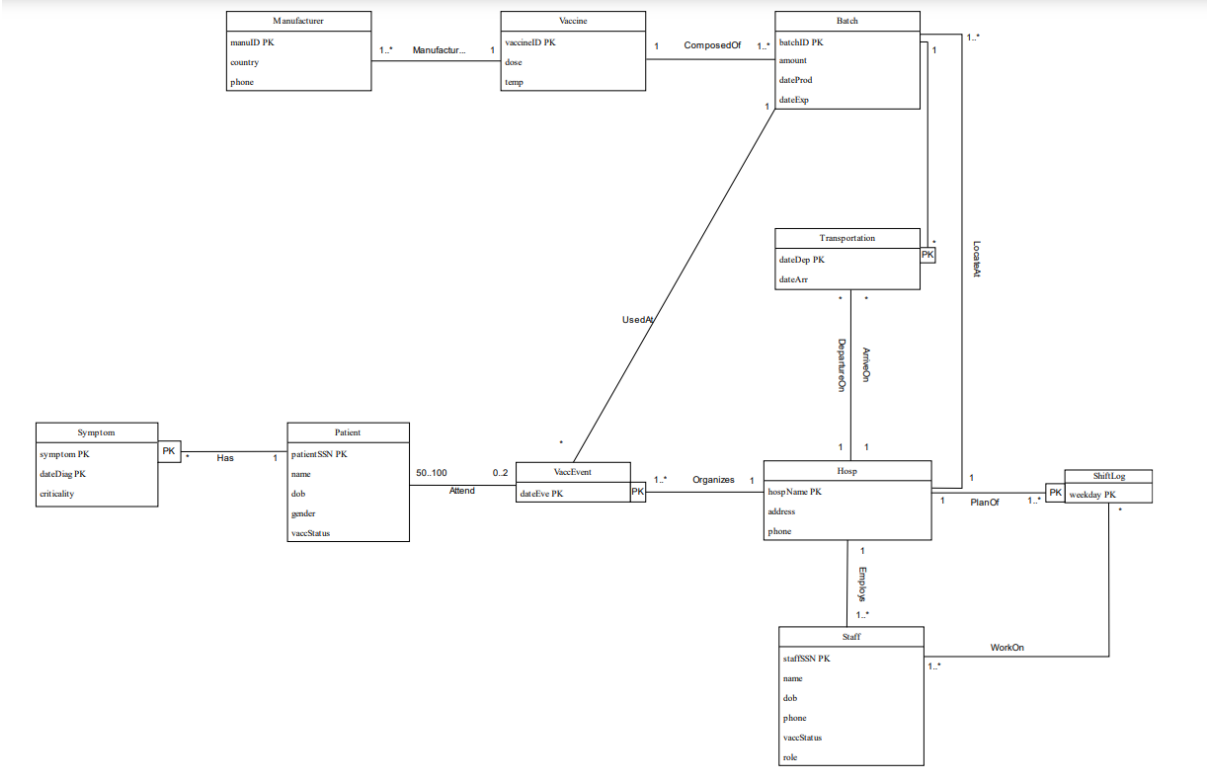

Figure 1first UML diagram

Revised UML diagram

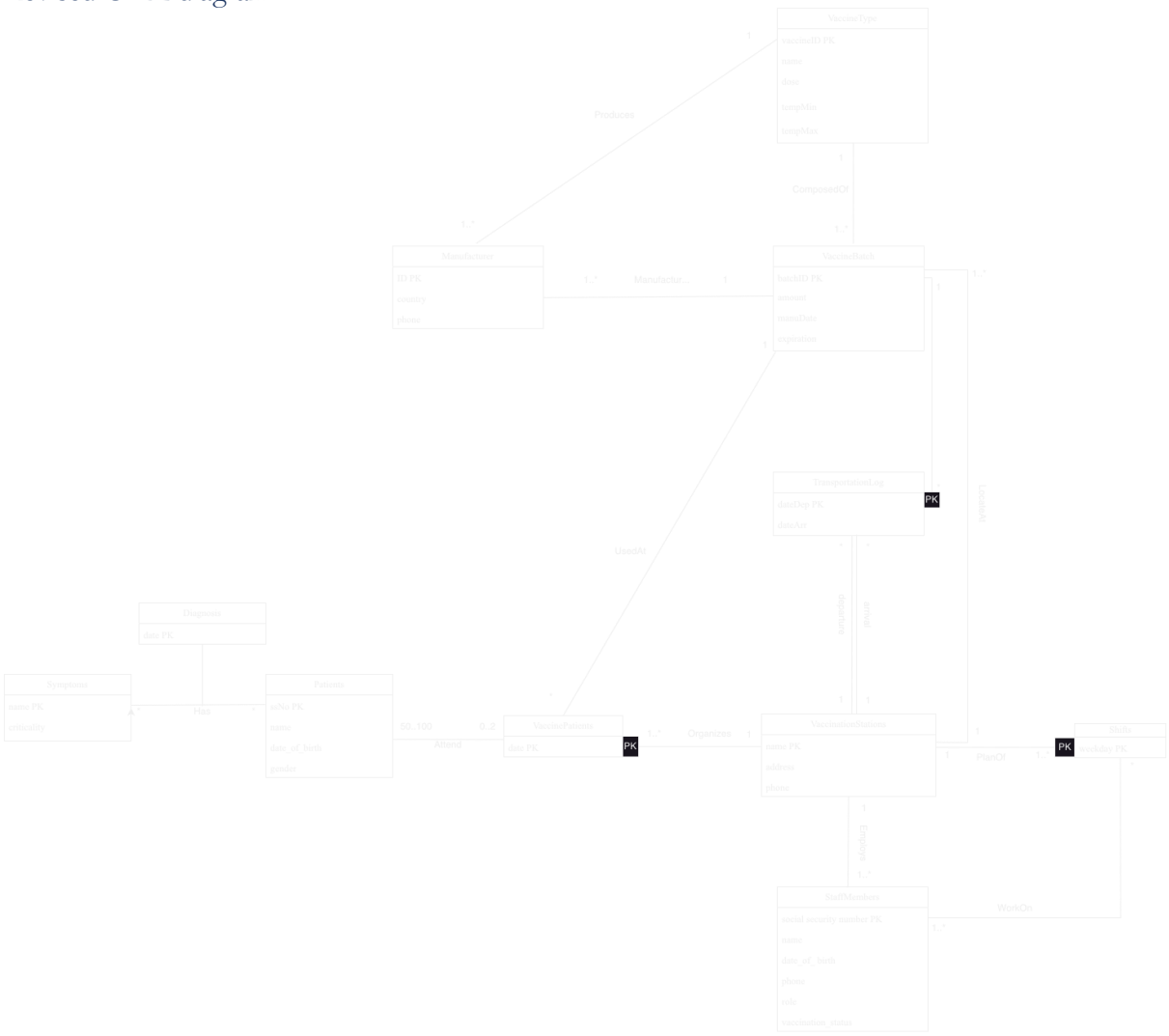

Figure 2 revised UML diagram

For light mode:
https://drive.google.com/drive/folders/1FJSVrwYSCG6U1wK09qZgcqoLsXg8PmkP?usp=sharing

First relation schema

Manufacturer(manuID, country, phone, vaccineID)
Vaccine(vaccineID, dose, temperature)
Batch(batchID, amount, dateProd, dateExp, hospName , vaccineID)
Hosp(hospName, address, phone)
Transportation(batchID, dateDep, dateArr, hospDep, hospArr)
Staff(staffSSN, name, dob, phone, vaccStatus, role, hospName)
ShiftLog(hospName, weekday)
WorkOn(staffSSN, weekday, hospName)
VaccEvent(dateEve, hospName, batchID)
Patient(patientSSN, name, dob, gender, vaccStatus)
Attends(patientSSN, dateEve, hospName)
Symptom(symptom, patientSSN, dateDiag, criticality)

Revised relation schema

Manufacturer(ID, country, phone, batchID)
VaccineType(vaccineID, name, dose, tempMin, tempMax)
VaccineBatch(batchID, amount, manuDate, expiration, name , vaccineID)
VaccinationStations(name, address, phone)
TransportationLog(batchID, dateDep, dateArr, nameDep, nameArr)
StaffMembers(social_security_number, name, date_of_birth, phone, role, vaccStatus, name)
Shifts(name, weekday)
WorkOn(social_security_number, weekday, name)
VaccinePatients(date, name, batchID)
Patients(ssNo, name, dob, gender)
Attends(ssNo, date, name)
Symptom(name, criticality)
Has(ssNo, name, date)

SQL files and usage

In the project, PostgreSQL relational database management system is used to create, manage, and manipulate the database. SQL queries are performed in creating tables and retrieving data to perform data analysis.

All SQL files are stored in database folder under 'cs-a-1155-2023-vaccine-distribution-group-17' folder. There are all together 11 sql files. The sql file 'create_and_file_db_psql' is used to create tables, and the file 'query1' to 'query7' are the queries answering project part 2 questions. SQL file 'vacc_patient', 'freq1', 'ferq2' are queries creating views. These 3 views should be created before running 'query7' as these views are the foundation of the analysis.

Assumptions

- Each manufacturer only works from 1 country.
- Each manufacturer can only make one type of vaccine.
- All phone numbers are unique, and no two individuals/manufacturers share a number.
- SSNs are unique.
- Phone numbers and SSNs cannot be NULL.
- Any transportation of vaccines must take at least 1 day.
- Vaccine batches can't be split
    - every vaccine in a vaccine batch must be at the same hospital.
- All staffs must be working at exactly 1 establishment.
- All hospitals and clinics must organise at least 1 vaccination event.
- Hospitals and clinics must have at least 1 staff.
- Staff can work multiple days, and there must be at least one staff member working when a vaccination shift is scheduled.
- Staff members are not required to work vaccination shifts (some staff members can work on no vaccination shift at all).
- No patient attends more than 2 vaccination events.
- Each vaccine type must be used to create at least 1 batch.
- Each vaccine type must have at least 1 manufacturer.
- Each batch must contain only 1 type of vaccine.

- Each hospital and clinic must have at least 1 vaccinations shift plan.
- A patient can be diagnosed with the same illness multiple times on different dates.
- A vaccination event must be hosted by exactly 1 hospital/clinic.
- Multiple patients can have the same symptoms on the same date.
- A patient can be diagnosed with multiple symptoms on the same date.
- There are no entries in ShiftLog when a vaccination event is not taking place.
    - For example, if Central Hospital holds vaccination events every Monday and Thursday, the primary key (Central Hospital, Tuesday) does not exist.

Design approach

When designing the database, we followed the following steps to ensure a well-structured and efficient database design.

We start by identifying the main entities presented in the database, such as vaccine batch, vaccine type, manufacturer, hospital and clinics, staff, patients etc.

Then we define attributes within each entity according to description. For example, for 'Manufacturer' entity, we defined attributes 'ID', 'Country', and 'Phone'. We also identify primary key for each class, such as 'ID' as primary key for 'Manufacturer' class, marked as 'PK'.

Then, we define the relations between entities. For example, one type of vaccine can be produced by many manufactures and one manufacturer should only manufactures one type of vaccine. We present these relations using lines with appropriate cardinality indicators, such as '1 to many' or 'many to many'.

Performance Analysis and improvements

Strong aspects of the database

• We have incorporated the provided data into the schema by taking the necessary data, process it and reformatting it to our needs. This has allowed us to run queries and perform data analysis without any significant problems

• The database is created is consistent with our ideas of the schema. This can be helpful in the long term if we want to improve the database in any ways as we already have a firm understanding of it.

• There is no redundancy existing in the database.

• All classes are in BCNF

Avenues for further improvement

In the future, several improvements could be made to the database. These developments can further enhance the database by increasing its processing speed, accuracy, versatility, and practical applicability in order to meet demands of real-world scenarios.

• Add triggers to ensure the consistency in location of the vaccine batches in TransportationLog and MedicalFacility (i.e., the last location of vaccine batches)

• Transaction to handle concurrent queries against the database

• Add Foreign Key constraints (e.g, ON DELETE SET NULL, ON UP DATE CASCADE)

Common query time
Time to execute query 1: 9.94 milliseconds.

Working as a team
Task division and roles
Throughout the project, we employed different configurations of work distribution, depending on the nature of the task at hand. Tasks that were easily parallelized were completed individually, while other tasks required execution as a whole team. One technique that seemed to work very well for us was to split the ownership over components of the database between the team members. Under this configuration, every team member is entirely familiar with the project, while each is specialized in more narrow subsets of the database features. We first used this approach during the analysis of the textual task description. Each team member was assigned a portion of the textual task and created a partial UML model based solely on the concepts discussed within their assigned section. Later on, we collaborated to integrate these partial models into a cohesive UML model, which served as the foundation for further work. This ownership distribution approach became our standard practice. We relied on the expertise of individuals specialized in specific classes or associations whenever we encountered problems within the database.
Another noteworthy example of effective work distribution occurred during the implementation of the PostgreSQL database, based on the relational schema. We identified two primary areas of work: table creation and data cleaning/table population. Table creation was fairly simple, so we had one team member working on this task. The other 2 members were in charge of data cleaning and table population, which were the more challenging and time-consuming tasks.
Although we frequently arranged in-person meetings, a significant portion of our work was facilitated by various online collaboration tools. Git served as the primary platform for contributing our respective sections of the database code, while Google Docs allowed us to collaboratively work on intermittent reports. Telegram was useful for scheduling meetings and sharing progress updates.

Estimated time schedule for group project
We worked mostly according to the deadlines in the course schedule. A more detailed timeline is demonstrated in the table below.

Week No	Tasks
1	Form group and create Telegram group chat for communication
2	Text analysis and UML Model (part 1)
3	UML Model, function dependencies and BCNF (part 2)
4	PostgreSQL, data cleaning, table creation
5	Table population and query creation
6	Group presentation
7	Group Project part 3
8 + 9	Final Deliverables

Problem solving as a group
Issues are bound to come up when working on a project, whether you're doing it individually or with a team. In our team, whenever we faced big problems, such as figuring out the schema or dealing with version control, we made sure to discuss it and come up with solutions together. This way, we all knew what difficulties we were dealing with and what was our plan to solve

them, which helped us make consistent decisions going forward. For minor issues like queries and data analysis, we either worked on it individually or in pairs. This way, we saved each other's time while still being able to share our own opinions on specific things.

Conclusion

Overall, we all agree that this project has been very enjoyable and successful.
While there are still rooms for improvement in our project, we are genuinely satisfied with the outcome we achieved. It was our first experience with a database project, and it provided us with valuable opportunities to familiarize ourselves with new tools like PostgreSQL and Git, which are highly beneficial and applicable in real-world scenarios. Additionally, our team consists of amazing members, each making valuable contributions to the development of this project.