Project Deliverable Package

Databases for Data Science

Group 18

15.06.2023

1. Introduction

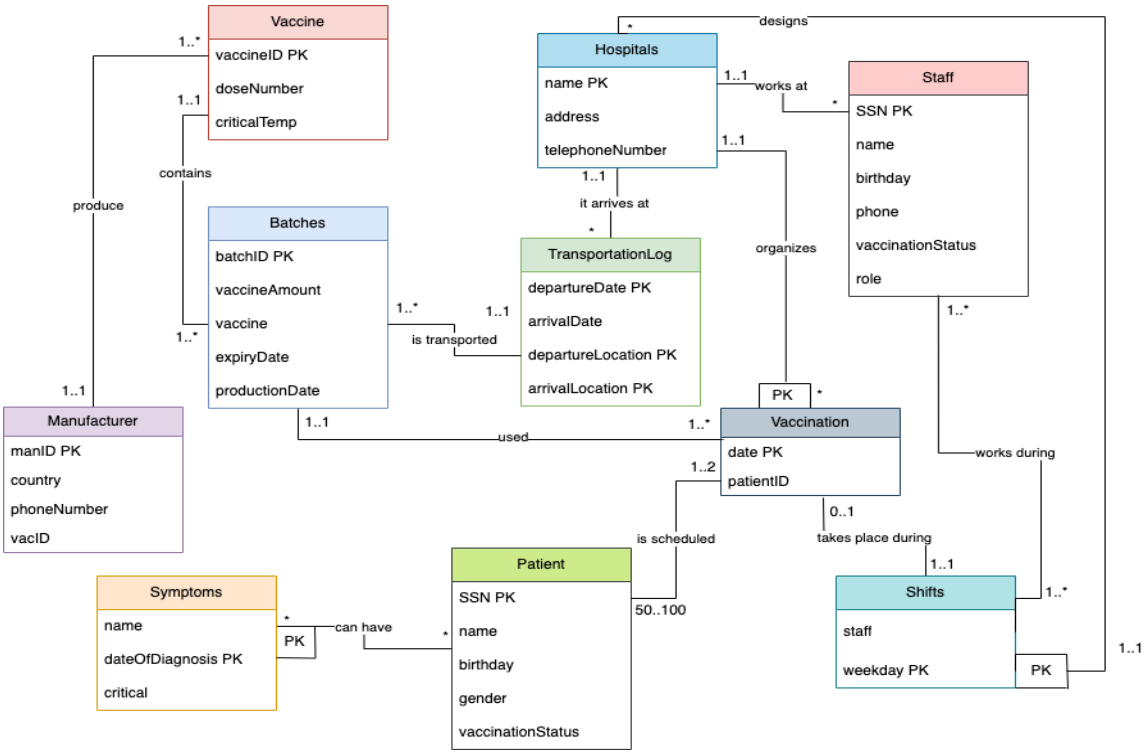
In this project, our team has created a database to keep track of the Covid-19 vaccine distribution and treatment in Finland. It tracks different vaccine types, transportation of vaccine batches, treatment plans, as well as patient information. In this document, we present the details of the project. The Python and the SQL files used to build this database and to run sample queries are included in the zip package together with this document itself. The files can be run by simply compiling the Python files or executing SQL queries.

2. Design and use cases

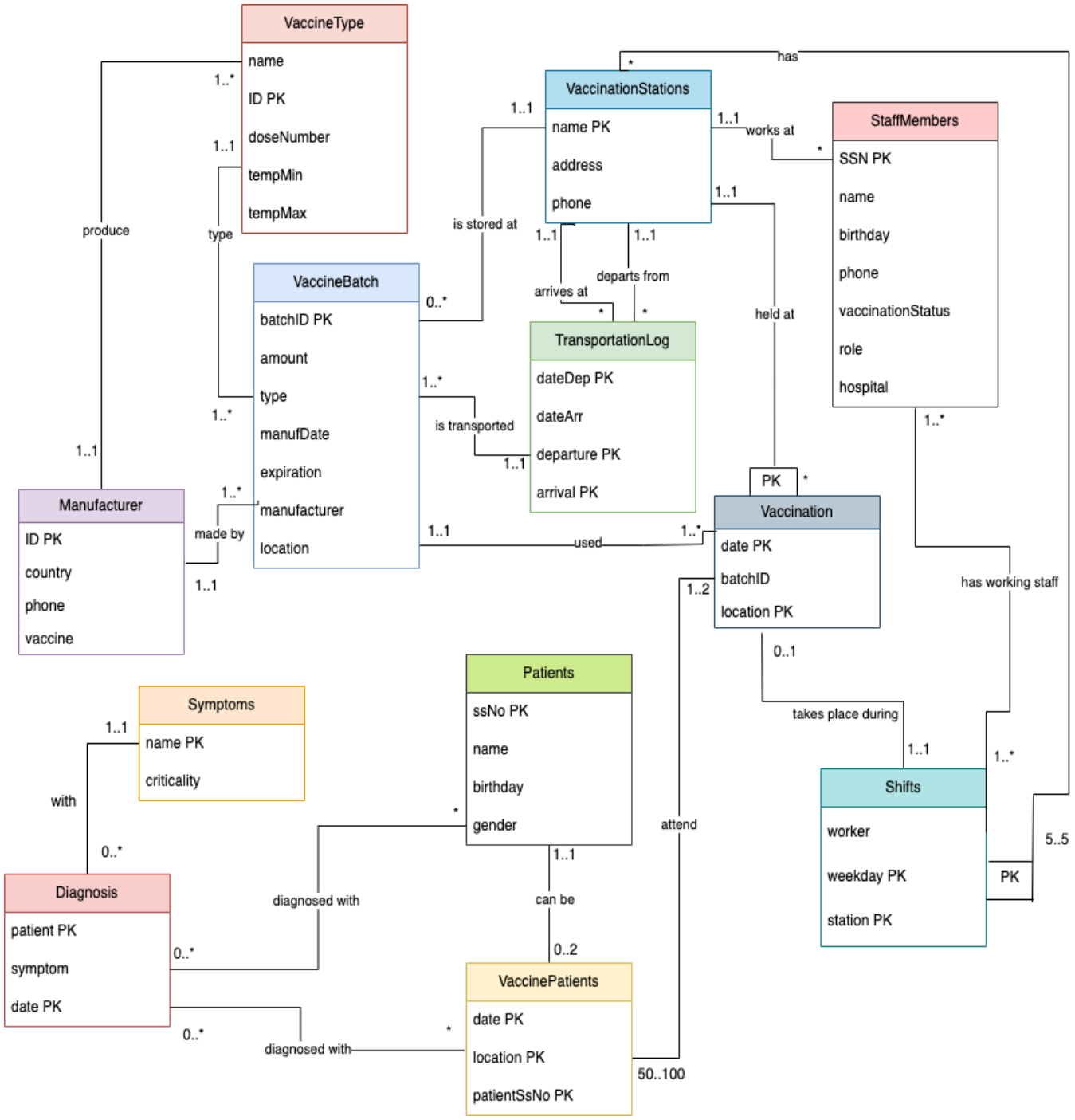
Use cases of the database

In the Python and SQL files included in the submitted zip package, there are also queries that demonstrate the possible uses of the database and some fundamental data analysis. For example, we have created a data frame to highlight the frequency of symptoms by vaccine type and gender, as well as keep track of the total number of vaccinated people by date. The database allows room for further analysis and deduction, hence, enabling users to derive useful information from the data, such as trends and correlation.

Initial UML diagram



Improved UML diagram



Improved relational databases

VaccineType(name, ID, doseNumber, tempMin, tempMax)
Manufacturer(ID, country, phone, **vaccine**)
VaccineBatch(batchID, amount, **type**, expiration, manufDate, location)
TransportLog(dateDep, dateArr, departure, arrival**)**
VaccinationStations(name, address, phone)
StaffMembers(SSN, name, birthday, phone, vaccinationStatus, role, **hospital**)
Shifts(weekday, station, **worker**)
Vaccination(date, location**, batchID, weekday**)
Patients(SsNo, name, birthday, gender)
Symptoms(name, criticality)
VaccinePatients(date, location, patientSsNo)
Diagnosis(patient, symptom, date)

Assumptions

- Not all patients from **Patients** are **VaccinePatients**.
- In **VaccinePatients**, one patient can be listed at most twice. A patient can only be listed at most twice because each vaccine type only needs the patient to be vaccinated with a maximum of 2 doses. When a patient is listed twice, it means that they have attended 2 vaccination events.
- **Diagnosis** involves patients from **Patients**. These patients may belong to VaccinaPatients or otherwise. One patient can have several diagnoses happening on different dates.
- Each staff member only works at one hospital/clinic.
- One manufacturer produces only one type of vaccine.
- There can be only one vaccination event per clinic per day.

Design choices

- We used various colors in our UML to increase the readability of the diagrams, since the colors help in distinguishing between tables.
- **Vaccination** and **Shifts** are matched by date with weekday, and location with station.
- UML design was changed to fit the given data to allow for easier input of data into the database.

3. Performance analysis and improvements

Possible improvements

Index Optimization: Analyzing query patterns and usage will help us identify frequently accessed columns for optimizing indexes. Proper indexing can improve our query performance by reducing the time required for the retrieval of data.

Query Optimization: Reviewing the SQL queries to minimize the use of wildcards and unnecessary joins which makes the queries more complicated.

Partitioning: We can partition large tables based on specific criteria, such as vaccineTypes and distribution regions (transportationLog). Dividing our data into smaller, more manageable chunks can enhance our query performance.

Normalization: We can evaluate the table structures to check if normalization can be applied to improve our data integrity and performance. Normalization can eliminate the data redundancies but could also require additional joins.

Effects on usability and efficiency

Schema changes were made to filter the wrong data from the **diagnosis** table due to incorrect date formats. This has positive effects on usability and efficiency, making the database more reliable and responsive.

By ensuring data accuracy and consistency, users can rely on the **diagnosis** table to make informed decisions. The improved data quality enhances usability, user trust and confidence in the system. Additionally, these changes improve the efficiency of data retrieval, hence providing faster query performance and reduced response times. Users can access relevant information more quickly, leading to a more efficient and user-friendly experience.

From an efficiency perspective, correct schema changes can optimize resource utilization by allocating CPU/memory space effectively. The usage of indexes on date columns becomes more efficient, further enhancing query performance.

4. Working as a team

Division of tasks

Part 1:
The entire group brainstormed the UML diagram together and we discussed the assumptions together. The UML diagram was completed by An Binh and Afrooz. Krzysztof and Ojaswi converted the UML diagram into the relational data model. For the non-trivial functional dependencies, again first the entire group discussed together. Then we divided the functional dependencies equally amongst ourselves and individually finished our part of the functional dependencies. Since there weren't many anomalies we again discussed it amongst each other and An Binh wrote it down.

Part 2:

Ojaswi and An binh finished the SQL tables along with the primary and foreign keys. Then Krzysztof and Afrooz reviewed it to make sure it was in accordance with part 1 of the project. We all agreed on what attributes should have the check constraints. This was fairly simple. Krzysztof populated the SQL tables with the sample data. Now we were left with 7 SQL queries. We decided to divide them and work on them independently and meet afterwards to discuss our solutions. Afrooz did query 1 and 3 part 1 and part 2. Ojaswi did the query 4 and An binh did query 2 and 5. Krzysztof did query number 7. We thought that the 6th query was a bit more challenging so we decided to do it together. It was by far the longest query. The pdf with the queries and answer sets were typed by Ojaswi and Krzysztof.

Part 3:

Since there were 4 harder exercises we decided to each take one hard exercise and divide the easier ones. The task divisions were as follows:
OJASWI - 1 AND 2 (difficult exercise)
AN BINH- 3 AND 7 (difficult exercise)
AFROOZ - 4, 5, AND 6 (difficult exercise)
KRZYSZTOF - 8,9 AND 10 (difficult exercise)

Then we formed pairs of two and reviewed each other's tasks. Ojaswi and Afrooz reviewed each other's tasks and An Binh and Krzysztof did the same.

Estimated schedule

Project part 1	Project part 2	Project part 3
Meeting 1 - 5th May	**Meeting 1 - 23rd May**	**Meeting 1 - 4th June**
Meeting 2 - 8th May	**Meeting 2 - 25th May**	**Meeting 2 - 7th June**
Meeting 3 - 14th May	**Meeting 3 - 29th May**	

Solving problems as a group

All of us in the team approached the problem with different methods for creating UML diagrams. So it was hard to decide which approach to take. After thorough discussion, we unanimously chose the more efficient methods. This decision streamlined our workflow, improved productivity, and enforced effective collaboration, ultimately leading to a successful resolution of the problem.

5. Conclusions

Advantages and Disadvantages of the database

Advantages:

- The model of the databases is fitted for easy input of data in the previously given format.
- The design is straightforward and easy to understand, which makes it easier to operate on the database.
- The database has very little repeating data and the repeated data is mostly used as foreign keys to link tables together.

Disadvantages:

- The Python file responsible for inputting data does not check every table because of lack of provided guidelines for data except from type.
- As stated previously in the efficiency section, the schema is not perfectly optimized but that could be improved in the future.

Future of the database

In the future, it is advised to try and overcome design problems described in the disadvantages of the database. Especially implementing the checks for correct data type is advised as this could prevent many possible problems in the future.

Overall evaluation

Overall this project was an interesting challenge. The deadlines were probably the hardest obstacle to deal with as there was quite a lot of work for such a short amount of time to be done. On the other hand, we have learned and benefited a lot from it.