# Vaccination Distribution Project - Final Deliverable Package

**Group 21**

Duy To

Minh Quang Tran

Thuy Duong Cao

The Hung Nguyen

# Table of contents

## I. Introduction

This is the documentation for the final deliverable package of the course CS - 1155 Databases for Data Science during Period V, Spring 2024 at Aalto University. This package includes this documentation, folder “code” containing all necessary SQL and Python files for creating the database, and folder “data” containing the data that will be used to populate the database.

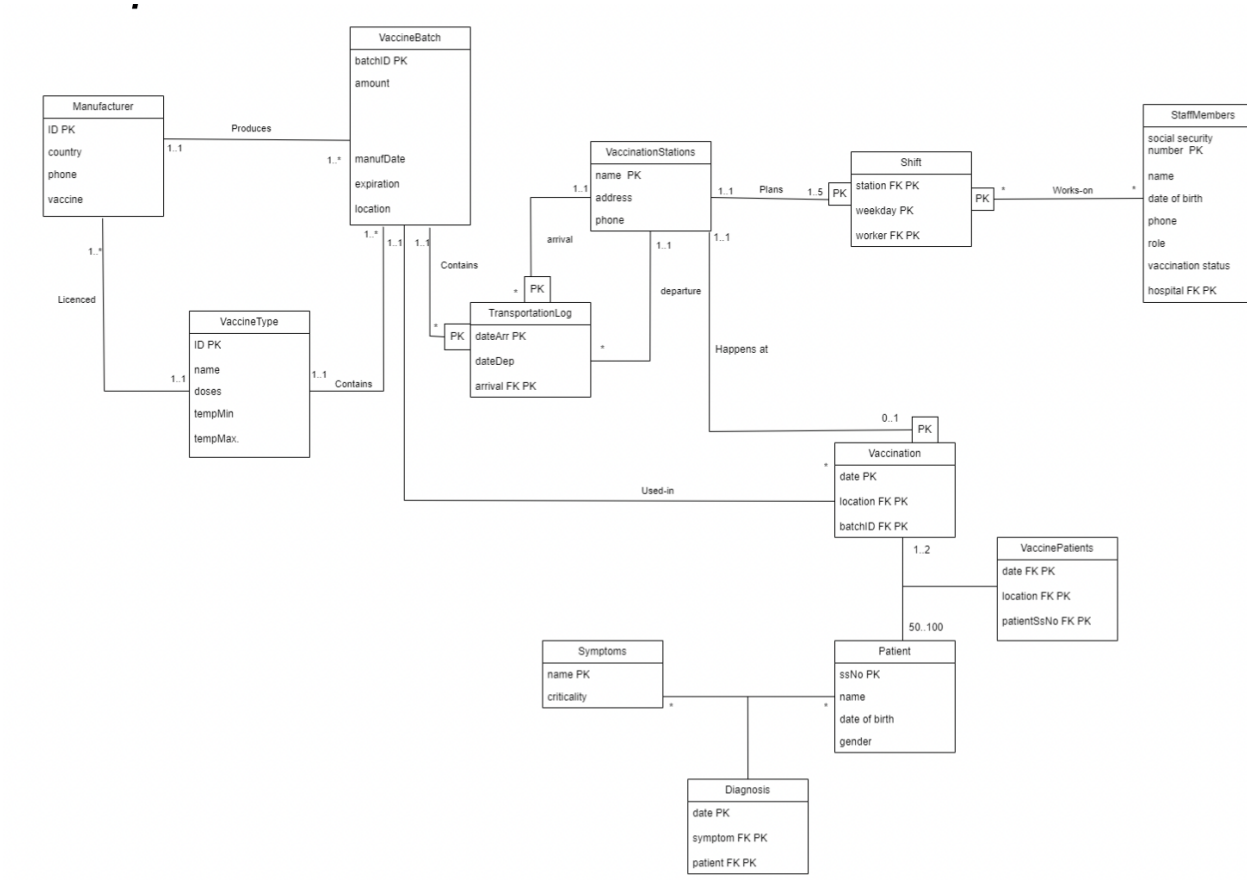
### Purpose of the database

The database stores information about vaccine types, manufacturers, transportation of vaccines, clinics and hospitals, staff at vaccine stations, vaccination events, patients, and diagnosis.

The database can be used to store new information and give insight into vaccine-related matters such as vaccination status of patients and the current location of a specific vaccine batch.

## II. Design and use cases

### 1. UML diagram and relational schema




Manufacturer(<u>ID</u>, country, phone, vaccine)

VaccineBatch(<u>batchID</u>, amount, type, manufacturer, manufDate, expiration, location)

VaccineType(<u>ID</u>, name, doses, tempMin, tempMax)

VaccinationStations(<u>name</u>, address, phone)

TransportationLog(<u>batchID</u>, <u>arrival</u>, departure, <u>dateArr</u>, dateDep)

Shift(<u>station</u>, <u>weekday</u>, <u>worker</u>)

StaffMembers(“<u>social security number</u>”, name, “date of birth”, phone, role, “vaccination status”, hospital)

Vaccination(<u>date</u>, <u>location</u>, <u>batchID</u>)

Patient(ssNo, name, “date of birth”, gender)

VaccinePatients(date, location, patientSsNo)

Symptoms(name, criticality)

Diagnosis(patient, symptom, date)

## 2. Instructions on using the files

To create the database using this package, which includes all the necessary sql files, follow these steps:

Step 1: In the package, find and open the file “test_postgresql_conn” in folder “code”

Step 2: Change the credentials such as username, password, host, and database.

Step 3: Make sure you have installed the required libraries in ./code/requirements.txt

Step 4: Run the python file.

## 3. Assumptions

- One worker can work on many weekdays
- One patient can receive at most 2 vaccines
- One batch of vaccine can be used in only one vaccination event
- To be considered a patient, a person must attend at least 1 vaccination event.
- One manufacturer produces only one type of vaccine.

## 4. Design choices

- In the final version, we decided to design our database according to the dataframe of the data we had in order to make populating the database easier.

## 5. Use cases

Some use cases:

- Estimate the amount of vaccines that should be reserved for each vaccination to minimize waste
- Plot the total number of vaccinated patients with respect to date
- Find the patients and staff an infected staff has met in 10 days.
- Find all the staff members working in a vaccination on May 10, 2021
- Find all patients with critical symptoms diagnosed later than May 10, 2021
- Find the total number of vaccines stored in each hospital and clinic
- For each vaccine type, you should find the average frequency of different symptoms diagnosed

## II. Analysis and changes

***The former vs the updated version of our relational schema***

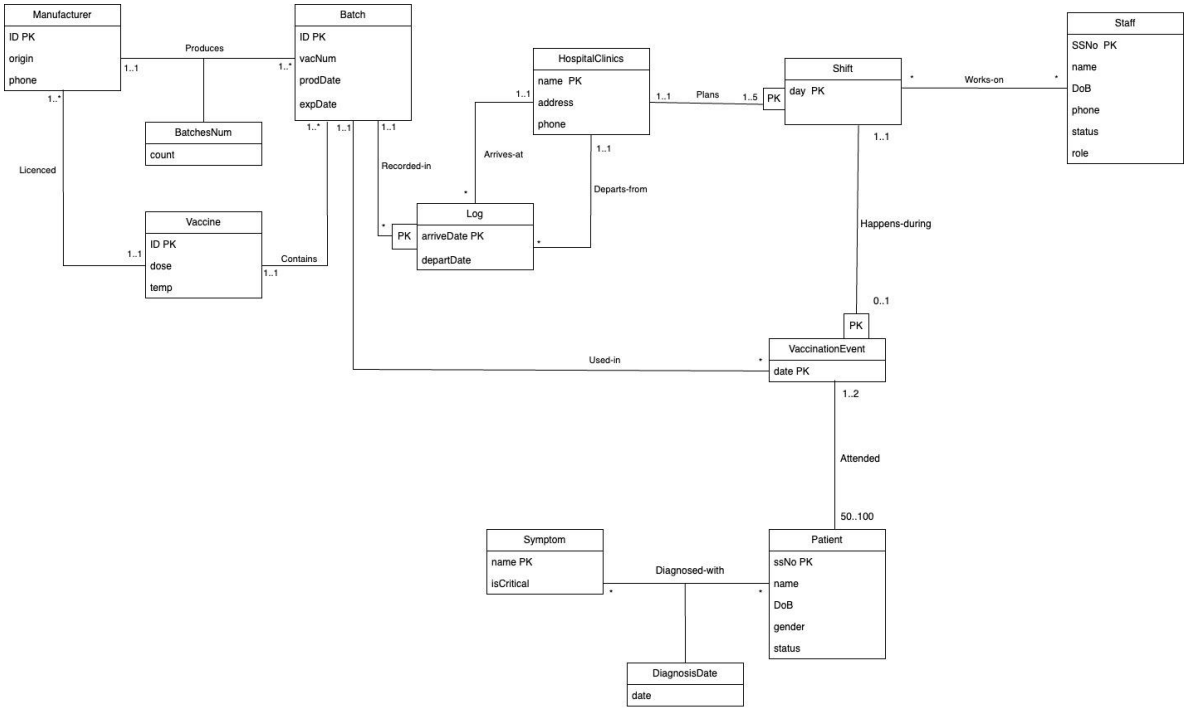
| Former | Latter |
|---|---|
| Manufacturer(ID, origin, phone, vaccine) | Manufacturer(ID, country, phone, vaccine) |
| Batch(ID, vacNum, prodDate, expDate, vaccineType, manuID) | VaccineBatch(batchID, amount, type, manufacturer, manufDate, expiration, location) |
| Vaccine(ID, dose, temp) | VaccineType(ID, name, doses, tempMin, tempMax) |
| HospitalClinics(name, address, phone) | VaccinationStations(name, address, phone) |
| Log(arriveDate, departDate, batchID, arrival, departure) | TransportationLog(batchID, arrival, departure, dateArr, dateDep) |
| Plans(hospitalClinics, shift) | Shift(station, weekday, worker) |
| Shift(hospitalClinics, day) | StaffMembers("social security number", name, "date of birth", phone, role, "vaccination status", hospital) |
| Staff(SSNo, name, DoB, phone, status, role) | Vaccination(date, location, batchID) |
| WorksOn(shift, staff) | |
| VaccinationEvent(shift, date, hospitalClinics) | Patient(ssNo, name, "date of birth", gender) |
| HappensDuring(event, shift) | VaccinePatients(date, location, patientSsNo) |
| Patient(ssNo, name, DoB, gender, status) | Symptoms(name, criticality) |
| Attended(hospitalClinics, shift, date, patient) | Diagnosis(patient, symptom, date) |
| Symptom(name, isCritical) | |
| DiagnosedWith(symptom, patient, diagDate) | |

Based on the newly given database, we have recognized that there could be some improvements to our relational model:

- · There were some renaming of relations and attributes in order to be compatible with the database. (eg. 'isCritical' -> 'criticality')
- · Old association relations 'Plans', 'WorksOn', 'HappensDuring' have been eliminated as they have been merged into other relations.
- · Relation 'VaccineBatch' now has an additional attribute 'location', which indicates the current location of the batch.
- · Attribute 'temp' of relation 'Vaccinetype' now is changed into 'tempMin' and 'tempMax' due to the database.
- · Relation 'TransportationLog' now has attributes 'batchID', 'arrival', 'dateArr' as primary keys instead of the first 2.
- · Relation 'Shift' now has an additional attribute 'worker' and now has all its attributes as primary key
- · Relation 'StaffMember' now has additional relatioes 'vaccination status' and 'hospital', as a result of merging attribute relations and following the database
- · Relation 'Vaccination' (previously known as 'VaccinationEvent') now has relation 'batchID' instead of 'shift'
- · relation 'Patient' now doesn't have attribute 'status' anymore, as vaccinated patients are now a different relation

- · Relation ‘VaccinePatients’ is now created according to the database, practically as an update of association relation ‘Attended’, just without attribute ‘shift’.

We proceed to update the UML model:




From this initial UML model:

- Classes and attributes are renamed
- class ‘Vaccination’ is now directly associated to class ‘VaccinationStations’
- There were additions to classes and attributes, according to the database.
- Foreign key was defined.

**III. Group work**

For each part, the roles were divided as follows:

Part 1:

Relational schema and UML construction: Duy To, Duong Cao, Quang Tran

Documentation and review: Hung Nguyen

Part 2:

SQL and documentation: Duong Cao, Quang Tran, Hung Nguyen

Python implementation and documentation: Duy To

Part 3:

No division of role, all members work together.

As a group, we solve a problem by means of online discussion hosting on Discord. Furthermore, we also make use of Gitlab for version control. We found Gitlab particularly useful to make sure everyone has access to the most updated version of the project, as well as maintaining uniformity.

## IV. Conclusion

### 1. Database evaluation

Advantages:

- Ease of implementation (the database can be easily set up following the instruction given in 2.3)
- Prevent faulty entries.
- Prevent anomalies, such as when one relation is modified, auto-cascade all related relations. (by using foreign keys)

Disadvantages:

- Lacking a GUI
- Efficiency when dealing with large databases is not tested.

### 2. Future improvement

- Building a GUI for ease of importing, exporting data, as well as plotting.

### 3. Overall assessment

- Overall, this database is suitable for the use cases stated in 2.5.
- It can be extended to cover more relations and use cases. For example, including information about doctors and treatment plans.
- Regarding teamwork, we as a group have worked harmoniously to solve all challenges posed by this demanding project.