COVID-19 VACCINATION REGISTRATION DATABASE

Mark Okafor and Catherine Nduasinde

April 2021

1 Introduction

An object-relational database (ORD) is a database management system (DBMS) that's composed of both a relational database (RDBMS) and an object-oriented database (OODBMS). ORD supports the basic components of any object-oriented database model in its schemas and the query language used, such as objects, classes and inheritance.

A hybrid database or Object-relational database it supports and uses both on-disk and in-memory data storage. Hybrid databases are used when the system needs high performance with the small footprint that only in-memory database systems can provide. This also provides the added benefits of durability and low cost of disk-based database systems. In short, the system makes use of hard disks for saving and retaining data, yet makes use of the memory for data that is in dynamic use to increase performance.

COVID- 19 VACCINE REGISTRATION DATABASE PROJECT was inspired on the current situation of spread of this pandemic disease, Although the vaccine has been administered but majority of people especially youth seems not able to register due to busy daily life schedule. This project will enable individual to set their vaccine appointment at their preferred time so as many people could get vaccinated as soon as possible.

The database is designed with fundamentals of Object-Relational database in which we use SQLAlchemy as the Python SQL toolkit and Object Relational Mapper that gives us the full power and flexibility of SQL since the ORM provided by SQLAlchemy sits between the SQLite database which is for storage and Python program and transforms the data flow between the database engine and Python objects.

Main primary users include Patient and vaccine administer. In our database all objects created include, Patient profile, Patient personal information address, type of vaccine requested, dates for appointment, site location, vaccine administer information will be able to be stored and retrieved when needed using Data Manipulation Language.

2 Methodology

Database schema
A database schema is the skeleton structure that represents the view of the entire database. It defines how the data is organized and how the relations among them are associated. It formulates all the constraints that are to be applied on the data. In this project we based on Logical Database Schema which defines all the logical constraints that applied on the data stored. Since we use Object-Relational database mechanisms, we created 11 Classes and its attributes which have functions or methods that's allow a user to put or access the objects stored.

E-R diagram
An entity relationship diagram (ERD) shows the relationships of entity sets stored in a database. An entity in this context is an object, a component of data. An entity set is a collection of similar entities. These entities can have attributes that define its properties. In this project ER diagrams symbols are used to elaborate the relationship of classes including using primary key and foreign key.

Data definition or Data description language (DDL)
Is a syntax for creating and modifying database objects such as tables, indices, and users. DDL statements are similar to a computer programming language for defining data structures, especially database schemas. In This project the syntax for table creation, data integrity constraints, table altering, and updating features will be illustrated.

3 Results

3.1 Schema for the Database

People

ID	Username	Password	Confirm_pw	Email	First_name	Last_name	DOB	SSN	Phone_No	Created_on

PersonalInfo

ID	People_ID	Address_ID

AddressType

ID	Entity

Address

ID	Address1	Address2	City	State	Zip	Country	Addr_type_ID

Vaccine

ID	Name	Manufacture	Ailment	Dose	Days_apart	Vaccine_type	Comments

Appointments

ID	PersonalInfo_id	VaccineAdmin_id	Appt_date	Sec_appt_date	Sec_appt_time	Location_id	Vaccine_id	comments

SiteLocation

ID	Site_name	Address_ID

PersonnelInfo

ID	Job_title	People_id	Address_id	WorkAddress_id

VaccineAdmin

ID	PersonnelInfo_id

Patient

ID	Personal_info_ID	Appointment_ID

Sessions

ID	Patient_ID	Appointment_ID	Vaccine_Admin_ID	Dose_rem

Figure 1: The Database Schema for Covid-19 Vaccination Registration Database

3.2 Entity-Relationship for the Database

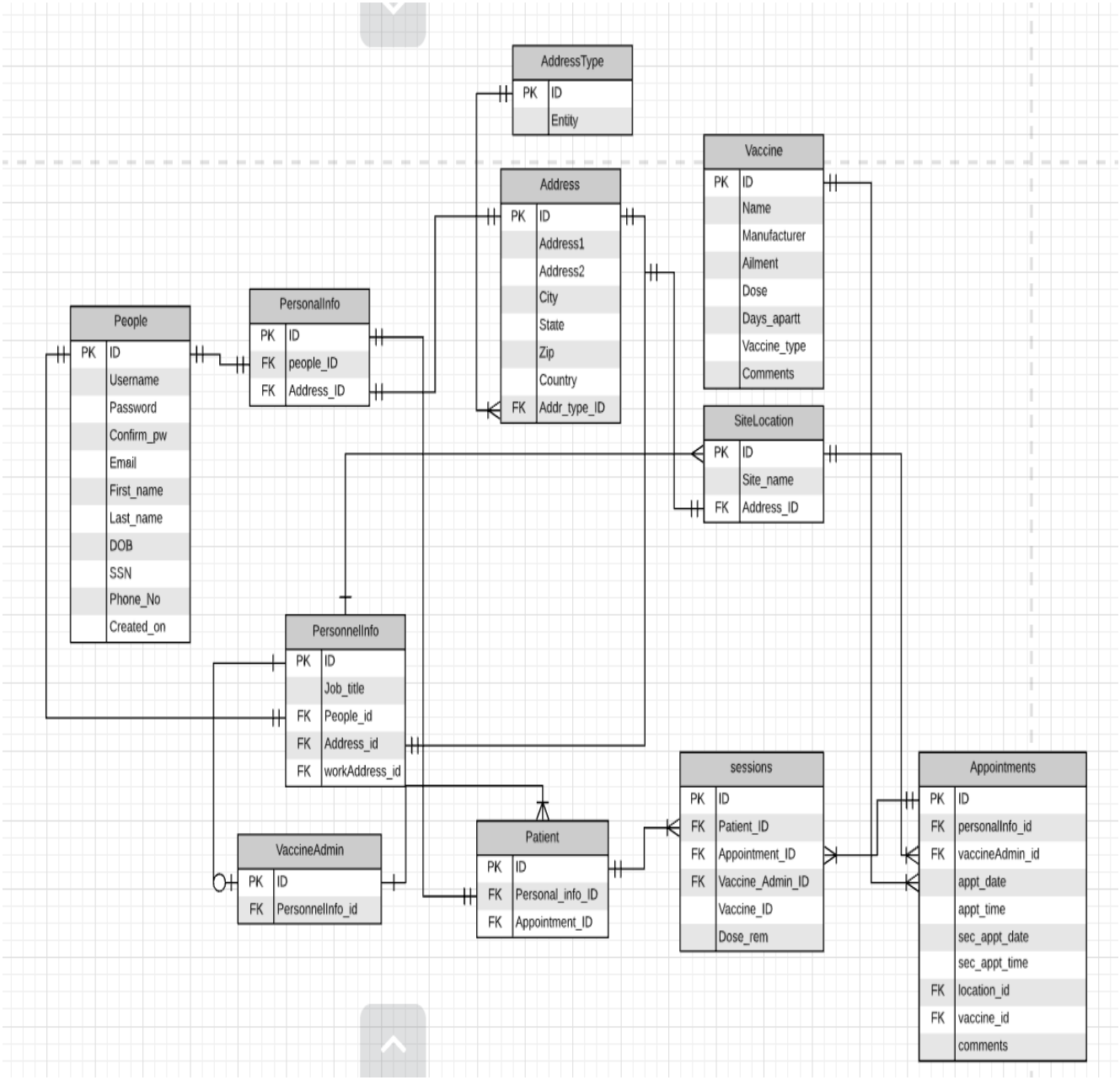


Figure 2: The Entity- Relationship diagram for Covid-19 Vaccination Registration Database

3.3 The Use data definition language (DDL) to create the database and constituent tables.

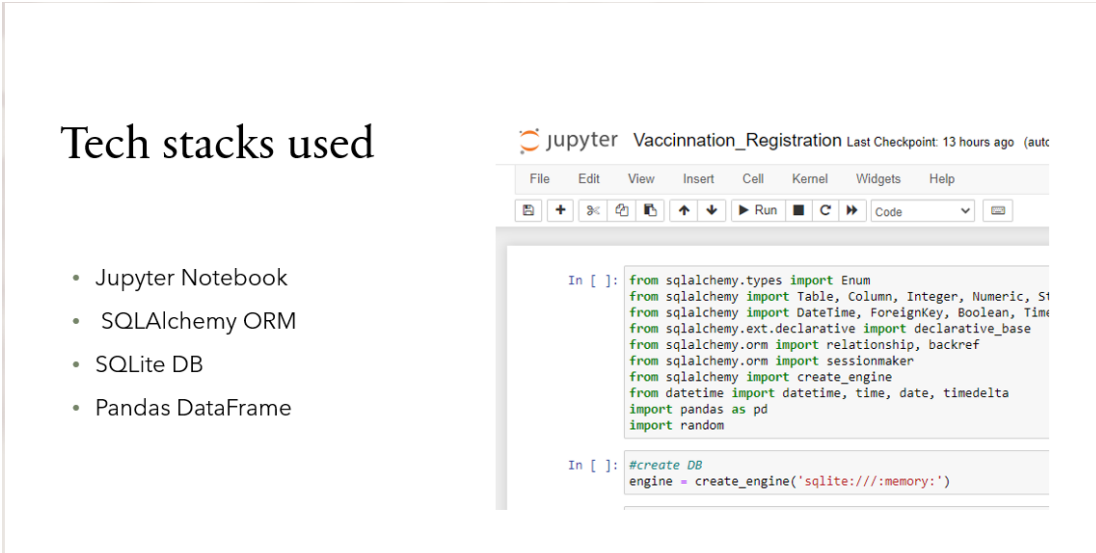


Figure 3: libraries and a command to create a database

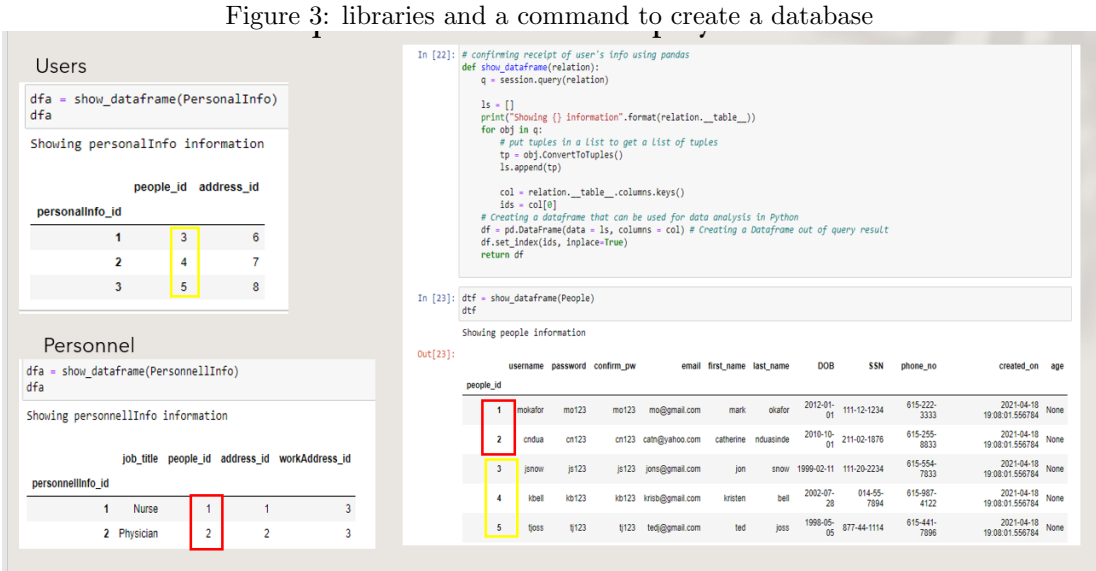


Figure 4: example of queries that can be performed

Class definitions using SQLAlchemy

CLASSES

- People
- AddressType
- Site
- PersonalInfo
- PersonelIInfo
- VaccineAdmin
- Vaccine
- Appointment
- Patient
- Sessions

```
In [ ]: class Vaccine(Base):
            __tablename__ = 'vaccine'
            __table_args__ = ({'extend_existing': True})

            vaccine_id = Column(Integer(), nullable= False, primary_key=True)
            name = Column(Enum('BNT162b2', 'mRNA1273', 'JN3786436735'), nullable= False, index=True)
            manufacturer = Column(Enum('PfizerBioNTech', 'ModernaTXInc', 'JohnsonAndJohnson'), nullable= False, index = True)
            ailments = Column(String(25))
            dose = Column(Integer(), nullable= False, index=True)
            days_apart = Column(Integer())
            vaccine_type = Column(String(32))
            comments = Column(String(255))


            def __init__(self, name, manufacturer, ailments, dose, days_apart, vaccine_type, comments):
                self.name = name
                self.manufacturer = manufacturer
                self.ailments = ailments
                self.dose = dose
                self.days_apart = days_apart
                self.vaccine_type = vaccine_type
                self.comments = comments


            def ConvertToTuples(self):
                tuples = (self.vaccine_id, self.name, self.manufacturer, self.ailments, self.dose, self.days_apart, self.comments)
                return tuples
```

Figure 5: Example of defining a Vaccine class using SQLAlchemy

3.4 Use CHECK and other integrity constraints during database definition.

3.5 INSERT a new row of data INTO any table

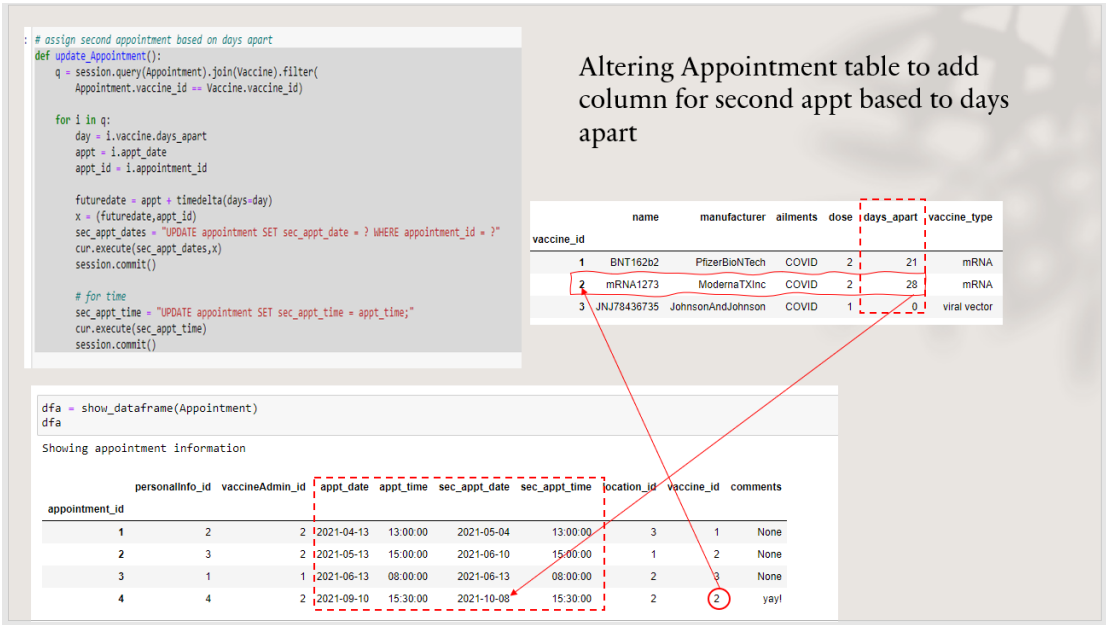


Figure 6: Example of inserting a new column

3.6 Updating a table

```
# creates a column for age on people relation
def create_age_column():
    updateColumn = "UPDATE people SET age = created_on - DOB;"
    cur.execute(updateColumn)
    session.commit()
```

```
dfa = show_dataframe(PersonalInfo)
dfa
```

Showing personalInfo information

personalInfo_id	people_id	address_id
1	3	6
2	4	7
3	5	8
4	6	9

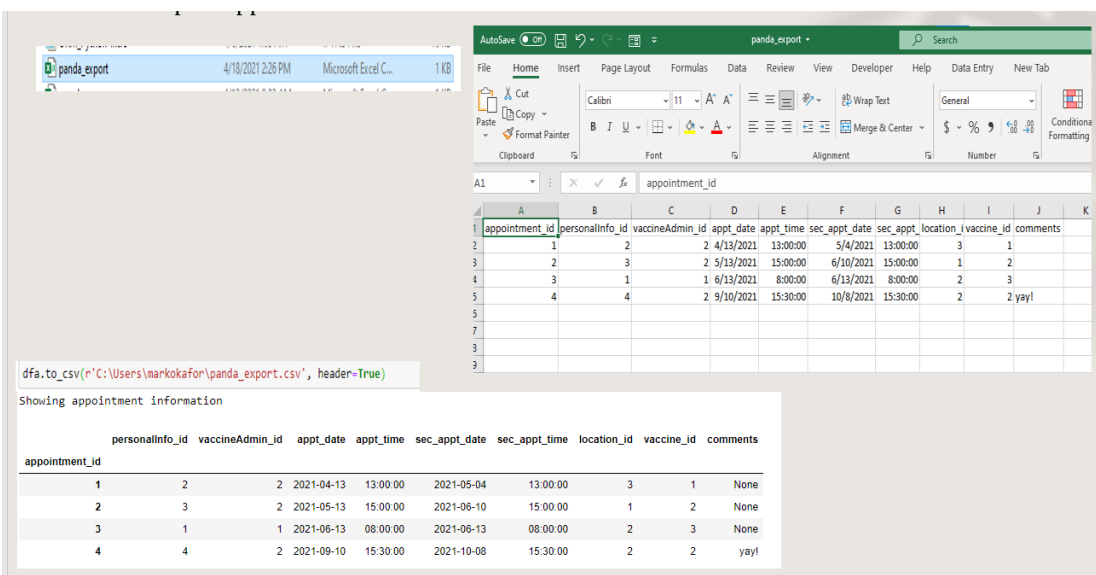
```
dtf = show_dataframe(People)
dtf
```

Showing people information

people_id	username	password	confirm_pw	email	first_name	last_name	DOB	SSN	phone_no	created_on	age
1	mokafor	mo123	mo123	mo@gmail.com	mark	okafor	2012-01-01	111-12-1234	615-222-3333	2021-04-18 19:08:01.556784	9
2	cndua	cn123	cn123	catn@yahoo.com	catherine	nduasinde	2010-10-01	211-02-1876	615-255-8833	2021-04-18 19:08:01.556784	11
3	jsnow	js123	js123	jons@gmail.com	jon	snow	1999-02-11	111-20-2234	615-554-7833	2021-04-18 19:08:01.556784	22
4	kbell	kb123	kb123	krisb@gmail.com	kristen	bell	2002-07-28	014-55-7894	615-987-4122	2021-04-18 19:08:01.556784	19
5	tjoss	tj123	tj123	tedj@gmail.com	ted	joss	1998-05-05	877-44-1114	615-441-7896	2021-04-18 19:08:01.556784	23
6	srogers	s123	s123	stever@gmail.com	Steve	Steve	1921-05-05	132-23-4567	623-20-8565	2021-04-18 14:24:02.143901	100

Figure 7: Altering the table to calculate the age of users

3.7 Importing external data in CSV or EXCEL format



```
dfa.to_csv(r'C:\Users\markokafor\panda_export.csv', header=True)
```

Showing appointment information

appointment_id	personalInfo_id	vaccineAdmin_id	appt_date	appt_time	sec_appt_date	sec_appt_time	location_id	vaccine_id	comments
1	2	2	2021-04-13	13:00:00	2021-05-04	13:00:00	3	1	None
2	3	2	2021-05-13	15:00:00	2021-06-10	15:00:00	1	2	None
3	1	1	2021-06-13	08:00:00	2021-06-13	08:00:00	2	3	None
4	4	2	2021-09-10	15:30:00	2021-10-08	15:30:00	2	2	yay!

Figure 8: Using Pandas to export appointment table to an excel file

4 Discussion and Conclusion

From this project we have learn how to create an Object- Relational database and we realize it can be a useful program for storing big data such as Information of Covid -19 Vaccinated individuals all over the country since there is no limitation on query complex request. The challenge for this project was time. As group we meet only twice a week for discussion, but we believe we could do more if we had more time. Our future work for this project would be to Implement user input validations Make it scalable, add more features, Apply more tools Build better relationship-type relations, Unique Constraints, Build using another tech stack and Add client User Interface by using Java script or CSharp.