**Database Project**

Christina Ye

hy2084@nyu.edu

**Part 1 Relational Database Design**

**Introduction**

In order to create a web-based system for registering covid-19 vaccination information and realize the functionalities such as registering patients, adding available appointments and offering appointments to patients, we first built up the schema for the databases and created the corresponding tables, some sample data were inserted to test the functionality of the model by applying the queries.

**(a) Relational Schema & E-R diagram**

Diagram

Description automatically generated**E-R diagram:**

**Relational schema:**

1. **Grouptype** (prionum,start\_date,description)

**Primary key:** (prionum)

**prionum:** number between 1-4 that specify the group a patient belongs to

**start\_date:** a date after which the patient in the group is eligible to get vaccination

**description:** the additional information for the group, such as age, career

1. **Patient** (patientId,name,SSN,birthday,address,latitude,longitude,phone,email,prionum,password)

**Primary key:** (patientId)

**prionum** references **Grouptype(prionum)**

**patientId:** it is unique and refers to the username of the patient in this system

**name,SSN,birthday,address,phone,email,:** patient’s information entered when registering

**latitude,longitude:** the coordination of the address, will be used to calculate distances

**prionum:** the group which the patient belongs to, will be derived by calculation and other documents

**password:** created when the patient registered, will be used to authenticate the patient when login

1. **Provider**(providerId,type,name,phone,address,latitude,longitude,password)

**Primary key:** (providerId)

**providerId:** it is unique and refers to the username of the provider in this system

**type:** the type of the provider, for example, it can be “hospital” or “pharmacy”

**name,phone,address:** provider’s information entered when registering

**latitude,longitude:** the coordination of the address, will be used to calculate distances

**password:** created when the provider registered, will be used to authenticate the patient when login

1. **Preference**(patientId,weekday,slot,distance)

**Primary key:** (patientId, weekday, slot)

**patientId** references **Patient(patientId)**

**patientId:** refer to the patient who enters this preference

**weekday,slot:** weekday will be numbers between 0-7, slot can be 8:00 or 12:00, which refers to time block 8:00am-12:00pm and 12:00pm-16:00pm

**distance:** the longest distance the patient can accept

1. **Appointment**(aid,adate,atime,providerId)

**Primary key:** (aid)

**providerId** references **Provider(providerId)**

**aid:** the id of the appointment, will be automatically generated when inserting new appointment

**adate, atime:** the date and start time of the appointment

**providerId:** the provider who entered the appointment

1. **Offer**(aid,patientId,offer\_date,deadline,reply\_date,status)

**Primary key:** (aid, patientId)

**aid** references **appointment(aid)**

**patientId** references **patient(patientId)**

**aid,patientId,offer\_date:** means that the appointment with the aid was assigned to patient with patientId on offer\_date

**deadline:** the latest date the offer can be accepted, if the offer was not replied after this date, it will be marked as “declined”

**reply\_date:** the date the patient reply to this offer

**status**: can be “unconfirmed”, “accepted”, “declined, “cancelled”, “missed” and “completed”. The offer can be cancelled after it is being accepted; The offer can be marked as “missed” if patient did not show up and “completed” if the appointment is completed, in other words, the patient received the vaccination

**Functionalities of the system:**

1. **Manage account**

The patients and providers can register accounts using different links, their account information and other information such as name, address will be stored in “patient” and “provider” tables.

1. **Add appointment & add preference**

After a provider login to its own account, it can add appointments; while the patient can login to his/her account to add a slot preference. The new added appointment and preference will be stored in “appointment” and “preference” table

1. **Assigning appointment**

The administrator can run algorithms to assign the available appointment to patients according to their preference and current availabilities, the assignment records will be stored in “offer” table

1. **Update offer**

When the appointment is assigned, the status will be initialized to “unconfirmed” until the patient replies.

When a patient accept an appointment, the status of this “offer” change to “accepted”;

When a patient decline an appointment, the status of this “offer” change to “declined”;

When a patient did not respond within deadline, the status of this “offer” change to “declined”;

When a patient cancel the appointment after the accepting, the status of this “offer” change to “cancelled”;

When a patient did not show up, the status of this “offer” change to “missed”;

When a patient received vaccination, the status of this “offer” change to “completed”;

**(b) Create database using MySQL**

**Create table statements:**

DROP TABLE IF EXISTS offer;

DROP TABLE IF EXISTS appointment;

DROP TABLE IF EXISTS preference;

DROP TABLE IF EXISTS provider;

DROP TABLE IF EXISTS patient;

DROP TABLE IF EXISTS grouptype;

CREATE TABLE grouptype(

prionum INTEGER primary key,

startdate DATE,

description VARCHAR(50)

);

CREATE TABLE patient(

patientId VARCHAR(20) primary key,

name VARCHAR(20),

SSN VARCHAR(9),

birthday DATE,

address VARCHAR(50),

latitude FLOAT,

longitude FLOAT,

phone VARCHAR(11),

email varchar(30),

prionum INTEGER,

password VARCHAR(20),

FOREIGN KEY (prionum) REFERENCES grouptype(prionum)

);

CREATE TABLE provider(

providerId VARCHAR(20) primary key,

providertype VARCHAR(10),

name VARCHAR(50),

phone VARCHAR(11),

address VARCHAR(50),

latitude FLOAT,

longitude FLOAT,

password VARCHAR(20)

);

CREATE TABLE preference(

patientId VARCHAR(20),

weekday INTEGER,

slot TIME,

distance INTEGER,

PRIMARY KEY (PatientId, weekday, slot),

FOREIGN KEY (PatientId) REFERENCES patient(PatientId)

);

CREATE TABLE appointment(

aid INTEGER NOT NULL AUTO\_INCREMENT,

adate DATE,

atime TIME,

providerId VARCHAR(20),

PRIMARY KEY (aid),

FOREIGN KEY (providerId) REFERENCES provider(providerId)

);

CREATE TABLE offer(

aid INTEGER,

patientId VARCHAR(20),

offerdate DATE,

deadline DATE,

replydate DATE,

status VARCHAR(10),

PRIMARY KEY (aid, patientId),

FOREIGN KEY (aid) REFERENCES appointment(aid),

FOREIGN KEY (patientId) REFERENCES patient(patientId)

);

The sample data are loaded through MySQL workbench.

**Example data after the insertion:**

select \* from p.grouptype

Table

Description automatically generated

Table

Description automatically generated with medium confidenceselect \* from p.patient

select \* from p.provider

A picture containing table

Description automatically generated

Table

Description automatically generatedselect \* from p.preference

Table

Description automatically generatedselect \* from p.appointment

Table

Description automatically generated

**Graphical user interface, table, Excel

Description automatically generated**select \* from p.offer

**(c) SQL queries**

(1)

INSERT INTO p.patient value("john98","John Smith","123456789","1999-01-01","1000 University Ave, New York, NY",40,-70,2222222222,"jacksmith@gmail.com",4,"123123")

Graphical user interface

Description automatically generated with medium confidence

(2)

INSERT INTO p.appointment(adate,atime,providerId) value("2022-01-01","12:00","flushinghospital")

Graphical user interface, application, table, Excel

Description automatically generated

he appointment is stored and it got an automatically generated aid “56”

(3)

with t1 as(

select aid,adate,atime,providerId,slot,patientId,distance

from test.preference,test.appointment

where weekday(adate)=test.preference.weekday and

TIMESTAMPDIFF(hour, slot, atime) >= 0 and

TIMESTAMPDIFF(hour, slot, atime)<4 and

adate > "2021-03-10" and

patientId="michael6"),

t2 as (

select aid, adate, atime, t1.providerId, t1.patientId, distance,

111.111 \* 0.62

\* DEGREES(ACOS(LEAST(1.0, COS(RADIANS(test.provider.latitude))

\* COS(RADIANS(test.patient.latitude))

\* COS(RADIANS(test.provider.longitude - test.patient.longitude))

+ SIN(RADIANS(test.provider.latitude))

\* SIN(RADIANS(test.patient.latitude))))) AS distance\_in\_miles

from t1, test.provider, test.patient

where test.provider.providerId=t1.providerId and

test.patient.patientId=t1.patientId)

select patientId, providerId, aid, adate, atime, distance\_in\_miles from t2

where distance\_in\_miles <= distance and

aid not in (select aid from test.offer where status="accepted" or status="completed" or status="unconfirmed")

order by distance\_in\_miles

Table

Description automatically generated

The first table “t1” joins appointment with patient’s preference, it selects all of the appointment according to the weekday and time in the preference, and it also filtered out the date before “2021-03-10” assuming this is the current date and we want appointment after this date;

The second table “t2” joins t1 with patient and provider, it also calculate the distance between the patient and provider;

Finally we select the appointments that are within the distance preference and are not “unconfirmed”, “accepted” or “completed”, the final result is sort by patientId and distance.

(4)

with scheduled as (

select distinct p.offer.patientId, prionum

from p.offer,p.patient

where status="accepted"

and p.offer.patientId=p.patient.patientId),

received as (

select distinct p.offer.patientId, prionum

from p.offer,p.patient

where status="completed" and

p.offer.patientId=p.patient.patientId),

waiting as (

select p.patient.patientId, prionum

from p.patient

where p.patient.patientId not in

(select patientId from scheduled union select patientId from received)),

scheduled\_num as (

select prionum, count(\*) as scheduled\_count

from scheduled group by prionum),

received\_num as (

select prionum, count(\*) as received\_count

from received group by prionum),

waiting\_num as (

select prionum, count(\*) as waiting\_count

from waiting group by prionum)

select p.grouptype.prionum,

coalesce(scheduled\_count, 0) as scheduled\_number,

coalesce(received\_count, 0) as received\_number,

coalesce(waiting\_count, 0) as wating\_number

from p.grouptype

left join scheduled\_num on p.grouptype.prionum=scheduled\_num.prionum

left join received\_num on p.grouptype.prionum=received\_num.prionum

left join waiting\_num on p.grouptype.prionum=waiting\_num.prionum

Table

Description automatically generated

First three tables “scheduled”, “received”, “waiting” are created to select the patients who got the appointments that are “accepted”, “completed” with their group number and the patient who have not accepted offer or completed vaccination with their group number

The following three tables “scheduled\_num”, “received\_num”, “waiting\_num” count the number of patients in different priority groups

Finally we join grouptype with the numbers, the null values are substituted by “0”

(5)

select p.patient.prionum, p.patient.patientId, p.patient.name, startdate

from p.patient,p.grouptype

where p.patient.prionum=p.grouptype.prionum

Table

Description automatically generated

The result is derived by join grouptype and patient on their group number.

(6)

with t1 as(

select patientId, count(\*) as missed\_count

from p.offer

where status="missed"

group by patientId

having count(\*) >= 2),

t2 as(

select patientId, count(\*) as cancelled\_count

from p.offer

where status="cancelled"

group by patientId

having count(\*) >= 3)

select p.patient.patientId, name

from p.patient

where p.patient.patientId in (select patientId from t1)

or p.patient.patientId in (select patientId from t2)

Table

Description automatically generated

The first table t1 select the patientId and the times they missed accepted appointments, and those who have missed 2 or more were selected

The first table t1 select the patientId and the times they cancelled appointments, and those who have missed 3 or more were selected

Finally we select the patients whose id belongs to these two groups (“Jones” has missed 2 times and “Ron” has cancelled 3 times)

(7)

WITH t1 AS (

select providerId, count(\*) as ct

from p.offer,p.appointment

where status="completed" and

p.offer.aid=p.appointment.aid

group by providerId),

t2 as (select max(ct) as m from t1)

select p.provider.providerId, name, t1.ct

from p.provider, t1, t2

where p.provider.providerId=t1.providerId and t1.ct=m

Graphical user interface, text, application

Description automatically generated

Since when a patient is vaccinated, the corresponding offer will be marked as “completed”, table t1 filter out these “completed” offer and gets the corresponding provider information, and calculate each provider’s count

t2 selects the max number from t1, which is the “largest number of vaccination”

finally providers who has number of vaccination that is equal to the “largest number of vaccination” are selected

**Part 2 Web Application Design**

**(a) Revision of the Relational Schema**

The original relational schema from part 1 was already able to realize the function of the system. In order to show the effect of the date on the result, the appointment date and eligible date in the example data is updated so some groups are eligible to get the vaccination from today but some are still not. In addition, there are some constraints on some of the attribute, which will be realized by combining the web application:

1. When inserting into **appointment**, the **adate** must be at least 7 days after today
2. When inserting into or update **offer**, the **offerdate** must be at least 7 days before the appointment date; The **deadline** is exactly 7 days after the **offerdate**; The **replydate** must be no later than the **deadline** (otherwise the patient cannot change the status of this offer); If the **status** equal to “offer”, it can be updated to “declined” or “accepted”; If the **status** equal to “declined”, “cancelled” or “missed”, it cannot be changed to other status; if the **status** equal to “accept”, it can be updated to “cancelled” by patient, or “complete” or “missed” by provider.

**(b) Webpage Design**

**The Overall Structure of the Application**

**Diagram, text

Description automatically generated**

**Functionalities of the Webpage**

**Index**

Graphical user interface, text, application, email

Description automatically generatedIf the user (patient or provider) is not logged in, it will show the message of login/register/signup options. The registration uses GoogleMap API to get address’s longitude and latitude and store them in the database.

**Patient Guide**

Text

Description automatically generatedFor logged in patient, it will show options of viewing the offer, update their personal information, add or delete preferences or logout

**Viewing the offer and update status**

If the user choose to view his/her offer, a list of the appointments will be shown

Example of user “jj999”

SQL query:

select appointment.aid, provider.name, provider.phone, provider.address, 111.111 \* 0.62

\* DEGREES(ACOS(LEAST(1.0, COS(RADIANS(provider.latitude))

\* COS(RADIANS(patient.latitude))

\* COS(RADIANS(provider.longitude - patient.longitude))

+ SIN(RADIANS(provider.latitude))

\* SIN(RADIANS(patient.latitude))))) AS distance\_in\_miles,

adate, atime, offerdate, deadline, replydate, status from offer, appointment, provider, patient where offer.aid=appointment.aid and appointment.providerId=provider.providerId and offer.patientId=patient.patientId and patient.patientId= "jj999 "

Graphical user interface, text, application

Description automatically generatedTable

Description automatically generatedNote that the user cannot update the appointment if the status is cancelled/declined/missed; If the status is “offer” the user can accept/decline the offer:

Example SQL query:

(1) Check if the appointment is valid first

select status from offer where patientId="jj999 " and aid=41 and CURDATE() <= deadline

(2) update with corresponding action if valid

update offer set status="accept ", replydate=CURDATE() where patientId="jj999 " and aid=41

If the user choose to accept the offer the message will be shown:

**Graphical user interface, text, application, email

Description automatically generatedUpdate information**

The user will be able to update their information (partial information will be updated if only part of the field is filled)

(1) select name, ssn, birthday, address, latitude, longitude, phone, email, password from patient where patientId = "jj999 "

(2) the result variables will be replaced with filled content if the corresponding field is filled

(3) update patient set name=?, ssn=?, birthday=?, address=?, latitude=?, longitude=?, phone=?, email=?, password=? where patientId=?

Graphical user interface

Description automatically generated with medium confidence

**Update Preference**

**Table

Description automatically generated**The patient can view/delete/add preferences for their vaccination appointment.

SQL query:

(1) Check if the preference is already exist

select \* from preference where patientId = ? and weekday=? and slot=?

(2) Do the corresponding insertion/deletion

delete from preference where patientId=? and weekday=? and slot=?

**A picture containing table

Description automatically generated**insert into preference(patientId,weekday,slot,distance) values (?,?,?,?)

**Provider Guide**

For logged in provider, it will show options for viewing a summary of the appointment, all of the appointment and information of assigned appointment and filter options. The provider can choose to add an appointment.

**Example queries for summary for the provider:**

(1) The count for each status of appointments

with t1 as (select status from appointment, offer where appointment.aid=offer.aid and providerId="walgreen1"),

t2 as (select count(\*) as o\_ct from t1 where status="offer"),

t3 as (select count(\*) as a\_ct from t1 where status="accepted"),

t4 as (select count(\*) as d\_ct from t1 where status="declined"),

t5 as (select count(\*) as c\_ct from t1 where status="cancelled"),

t6 as (select count(\*) as m\_ct from t1 where status="missed"),

t7 as (select count(\*) as com\_ct from t1 where status="complete")

select o\_ct, a\_ct, d\_ct, c\_ct, m\_ct, com\_ct from t2,t3,t4,t5,t6,t7

(2) All appointment

select adate, atime from appointment where providerId="walgreen1" order by adate, atime

(3) Summary of the assigned appointment

Table

Description automatically generatedselect offer.patientId, offer.aid, adate, atime, patient.name, patient.phone, offerdate, deadline, replydate, status from offer, appointment, patient where offer.aid=appointment.aid and offer.patientId=patient.patientId and providerId="walgreen1"

**Graphical user interface, text, application

Description automatically generatedAdd New** **Appointment**

The web application will first check if the date entered is at least one week from today and then perform the insertion:

insert into appointment(adate, atime, providerId) values (?,?,?)

One provider can have multiple appointments at the same time, the appointment will be assigned an aid when inserting into table “appointment”.

**Filtering the “missed” appointment and sort them by appointment time:**

Table

Description automatically generatedselect adate, atime, patient.name, patient.phone, offerdate, deadline, replydate, status from offer, appointment, patient where offer.aid=appointment.aid and offer.patientId=patient.patientId and providerId="walgreen1" and status="missed" order by adate, atime

**Administrator Interface**

The Graphical user interface, text, application

Description automatically generatedindex page for the administrator:

The administrator will be able to check each patient’s information and change their priority

Table

Description automatically generatedselect patientId, name, ssn, birthday, address, phone, email, patient.prionum, startdate from patient left join grouptype on patient.prionum=grouptype.prionum

The administrator can click on “change” to update the priority group for the corresponding patient:

Graphical user interface, text, application, chat or text message

Description automatically generatedExcept from running the matching algorithm every hour, the administrator will be able to manually run algorithm to match the patients and appointments and send confirmation to the patients’ email. The details of the algorithm is stated in the follwing section.

Table

Description automatically generated

The corresponding user will receive the email:

Graphical user interface, text, application

Description automatically generated

**(c) Other implementation**

In order to guard against SQL injection and cross-site scripting attacks, every parameters which will be parsed into the SQL query is processed using htmlspecialchars functions. After user enter some variable into the input fields on the website, the variables posted will be processed by htmlspecialchars, and the variables in the address will also be parsed to this function.

For each user session, user’s authentication(such as username, user ID, and password) is stored in session variable. When the user access one webpage, the code will first examine whether the user is logged in and show the corresponding content according to the session variable. If the user is not logged in, a message that require user to login/register will appear on the website.

**(d) Matching algorithm**

1. The first table joins patients’ information with their group number to get the eligible date and join with preferences on patients’ ID number. It also joins with appointment and the results which have the appointments matches in weekday, time slot and at least a week from today are chosen.
2. The second table filtered out the appointment that has been offered/accepted/completed. In addition, if the patient already has an offered/accepted/completed appointment, he/she will also be filtered out and not participate in the following steps.
3. If the patient once cancelled/missed/declined an appointment, then this appointment will not be assigned to the same patient.
4. Calculate the distance from patient’s address to provider’s address, if the distance is greater than the corresponding preferred distance, it will also be filtered out.
5. For each of the appointment in the table we have for now, compute how many patient is competing for this appointment.
6. Sort the result by group priority number, the count of patients who are competing for this appointment, appointment date, appointment time and distance.
7. Insert the information in first row of the table
8. Repeat step 1-7 until there is no record in the table

**Explanation of the algorithm:**

For step 1~4, they are filtering the appointments that matches patients’ schedule, preferred distance and eligible date. The appointment date must be greater than the eligible date, at least one week from the assigned date (today), and is not currently assigned to any patient. In addition, if the patient currently is assigned (offered/accepted/complete) an appointment, he/she will not participate in this process, and the cancelled/declined/missed appointment will not be assigned to the same patient.

The result table is sorted by group priority number, the count of patients who are competing for this appointment, appointment date, appointment time and distance. The table is sorted in this sequence is that the patient with higher priority number will be considered to assigned the appointment. Then the table will sort the appointment by the count of patients who are competing for this appointment, for example, if a patient is eligible for appointment A and appointment B, and 1 person is competing for the appointment A while 2 person is competing for the appointment. If the patient is assigned appoint A, then there will be only one person who are competing for the appointment B. Therefore, sorting by the count of patients who are competing for this appointment can assigned the appointments which are competed by less patients first so the count of patients who compete for other appointment decreases. Finally the result is sort by appointment date, appointment time and distance.

After The first record in the table is fetched, the corresponding patient will be offered corresponding appointment, and he/she will not participate when we run the query again. Then we repeat the step above and fetch the first record each time until there is no record in the table. In other words, all patients who originally has not been assigned an appointment and is eligible for some appointment now is offered one appointment.