National Institute of Technology
Silchar

Patient Record Database Management System for a Quarantine Centre

Database Management System Mini-Project

Project submitted to:
Dr. Sameer Borgohain

Bachelor of Technology Mini-Project
Database Management System, DBMS
CS-302

Manali Gogoi

August 2022
Silchar, Assam

Contents

Chapter 1

Introduction

Although DBMS is not a new concept, it is a very important one, still possessing huge potentials to enhance and systematize big data in any form. To ease the burden on many complex tasks that we have today, DBMS still plays an important role in the management of big data required to solve such problem statements. Be it banking Systems, Quarantine system records, Traffic records, etc. DBMS has always helped serve its tedious purpose without humans having to ponder on these complexities. With the current situation of the pandemic menacing the world, it is of utmost importance for associations to come up with efficient Quarantine System Management tools using Databases, now more than ever. Although many hospitals and Quarantine Centers have been using a database for ages, maintaining an efficient DBMS system is still a very important task for any organization. We would hence be implementing such a Patient Record maintenance system for a Quarantine Facility, using a DBMS system that is very efficient in handling such large patient record datasets. This mini-project aims to hence build a system for maintaining patient records for a quarantine centre and ensure efficient storing and accessing of data. The project uses standard procedure for designing the database, that includes E-R modeling, conversion of E-R model to relational model, normalisation of relational tables and use of triggers and stored procedures to implement certain functionalities. It uses MySQL connected with Python to implement the project and will be applying normalisation (upto 3NF) for the same. The IDE used for development is PyCharm(2020 edition).

Chapter 2
Problem Statement

NIT Silchar management has handed over Hostel 1 and 2 to the district administration to be utilized as a quarantine center. Each of the quarantined buildings is multi-storied consisting of ground floor with 100 rooms, 1st floor with 100 rooms and 2nd floor with 50 rooms only. So, the total no. of rooms per quarantine centre is 250. The ground floor rooms will be allotted to senior citizens (above 60 yrs of age), 1st floor will be allotted to the persons in the age group of 40 yrs to 60 yrs and the 2nd floor will be allotted to persons below 40 yrs. Each quarantined person will be allotted a single room for his stay at the quarantine centre for the said period.

Subject to the availability of rooms in the quarantine centre, district administration allows a limited no. of passengers to the said facility. So, each day a limited no. of passengers are directed to the quarantine facility at NIT Silchar. The nodal officer who administers the quarantine centre has to keep track of the records of the quarantined persons like the name of the person, address, multi-valued mobile no, age, arrival date, coming from, going to, allotted Room no, discharged date etc. It is required by us to build a quarantine facility management system that provides an ecient way of storing and accessing the data recorded by the quarantine centre.

Chapter 3

ER Modelling

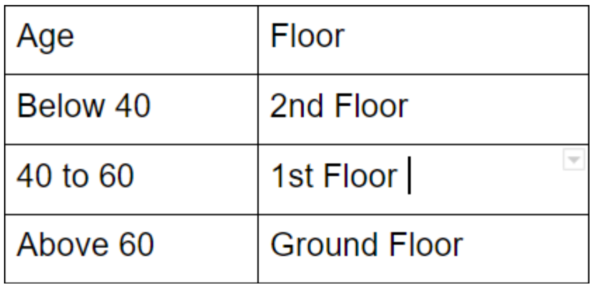
3.1 Entity Relationship

Entity Relationship model (ER model) is a high-level data model. It is used to define the data elements and relationship for a specified system. It develops a conceptual design for the database. It also develops a very simple and easy to design view of data. In ER modeling, the database structure is portrayed as a diagram called an entity-relationship diagram. An ER model is considered as a design or blueprint of a database that can later be implemented as a database. 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. A relationship shows the relationship among entities. By showing relationships among tables and their attributes, ER diagram shows the complete logical structure of a database.

3.2 Requirements of the database

Objective: The database is designed to store the details of the flight bound travellers without accompanying children who are being quarantined in NIT Silchar Hostel 1 and Hostel 2.

Specifications of allotment: The two hostels, i.e Hostel 1 and Hostel 2 are three storeyed RCC buildings consisting of 100 , 100 , 50 rooms in ground floor, 1st and 2nd floor respectively. The procedure for allotment is as follows :



Age	Floor
Below 40	2nd Floor
40 to 60	1st Floor
Above 60	Ground Floor

3.2.1 Requirements:

Allotting a single empty room to each quarantined person. Discharging a quarantined person after 14 days of admission. Querying for information related to a specific passenger or room.

3.2.2 Assumptions:

Room once allocated to a passenger will not be changed unless there is unavoidable circumstance. Passengers will be allotted in room in other floors if all the rooms in the allotted floor are occupied Passengers will be directed from Airport to NIT Silchar only if rooms are available on that day. Room Number follows the Format HFXX, where H = Hostel Number, F = Floor Number and XX = Room Number.

3.3 Entities

The entities for the following problem statement are as follows :

Traveler: The attributes are-

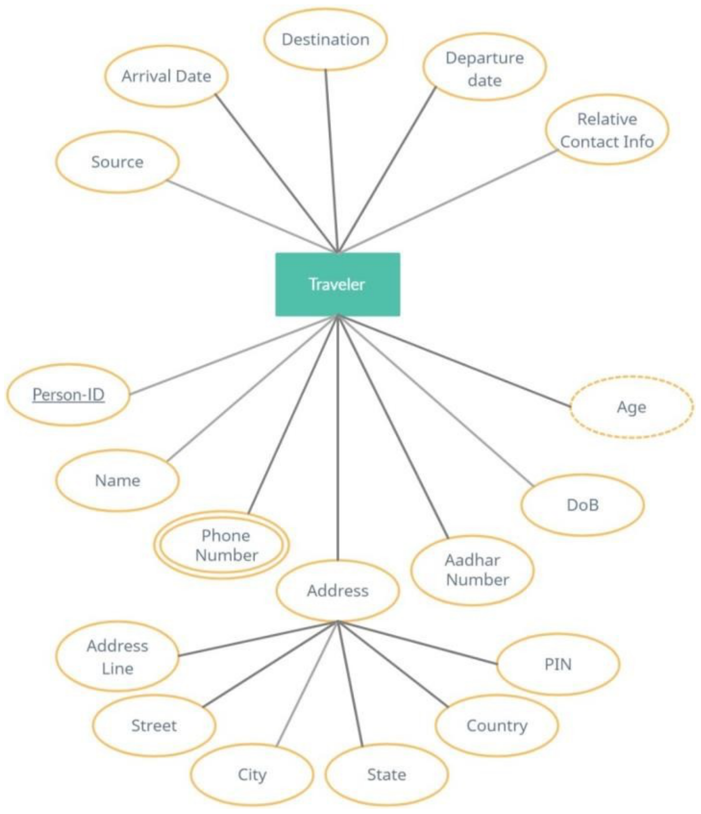
- Person-ID / Patient-ID
- Name
- Address
- Aadhar Number
- Phone Number
- DoB(Date of Birth)
- Age

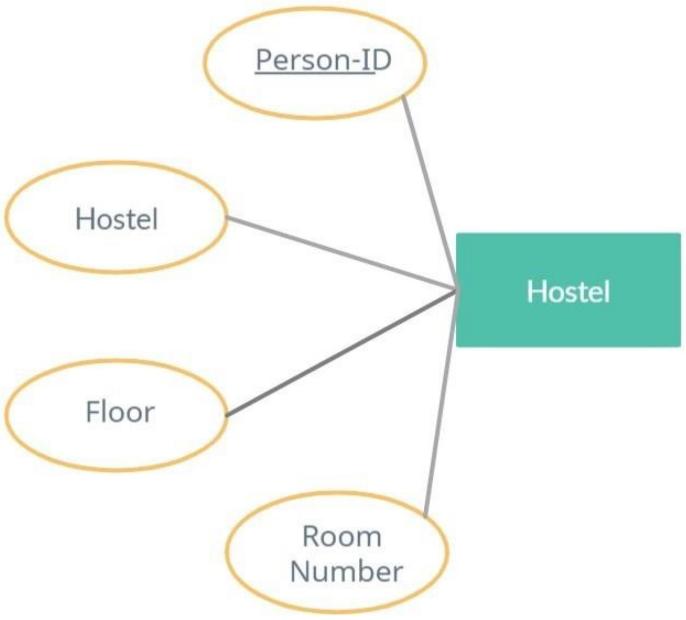
- Source
- Arrival date
- Destination
- Departure date
- Relative contact information

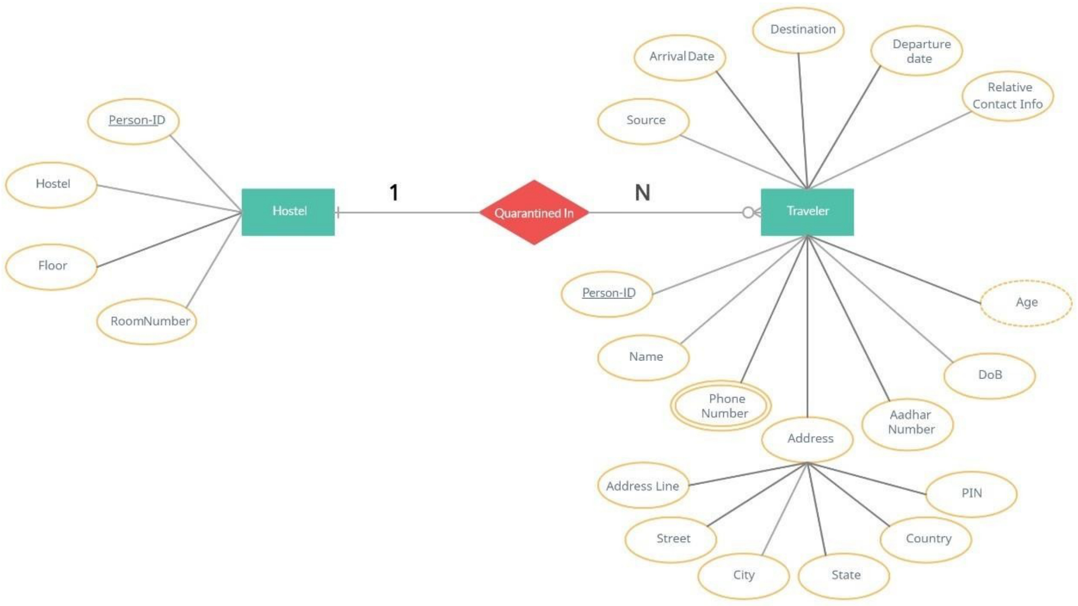
Hostel: The attributes are-

- Person-ID / Patient-ID
- Hostel Number
- Floor Number
- Room Number

3.4 ER Model Diagrams




Person-ID
Hostel
Floor
Room Number
Hostel


Person-ID
Hostel
Floor
RoomNumber
Hostel
1
Quarantined In
N
Traveler
Arrival Date
Destination
Departure date
Relative Contact Info
Source
Person-ID
Name
Phone Number
Address
Aadhar Number
DoB
Age
Address Line
Street
City
State
Country
PIN

3.5 Relationships between entities

The primary key for a quarantined person will be its Person-ID. This Person-ID will be assigned with the help of triggers. When A traveller arrives at the quarantine facility, a room will be allotted to him from the two hostels according to allotment policies. The relation between hostel and traveler will be expressed as “A traveller is quarantined in a Hostel”. The cardinality of this relation will be N : 1 because many travellers may be quarantined in one hostel.

The address of a patient comprises various fields such as Address line, Street, Area, City, State, Country, Pin code. The E-R diagram also represents the address as a composite Attribute. The E-R diagram represents “phone number” with double ellipses around it, meaning it is a multi-valued attribute. The Age attribute is a derived attribute as it can be derived from Date of Birth.

Chapter 4

Normalization Report

4.1 Functional Dependency:

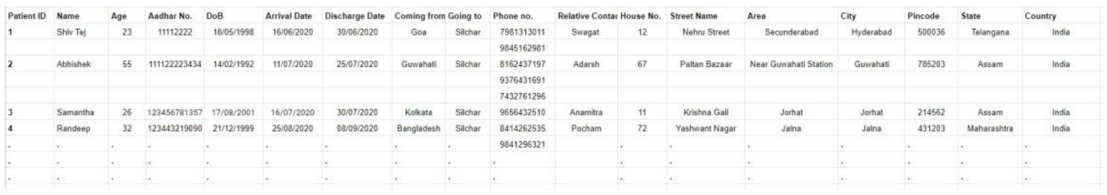
Functional dependency (FD) is a set of constraints between two attributes in a relation. Functional dependency says that if two tuples have same values for attributes A_1, A_2..., A_n, then those two tuples must have to have same values for attributes B_1, B_2, ..., B_n. Functional dependency is represented by an arrow sign (→) that is,

$$X \rightarrow Y$$

where X functionally determines Y. The left-hand side attributes determine the values of attributes on the right-hand side. Normalization is a method to remove all these anomalies and bring the database to a consistent state. The process of normalization helps remove redundancy in data provided by the patients. In the above problem statement, it can be observed that a patient may have multiple phone numbers. So, storing this will cause redundancy. Therefore, to remove redundancy from the database, normalization will be performed.

E-R model, when converted to Relational model, must be expressed through relational tables/ relational schema. The tables are shown below:

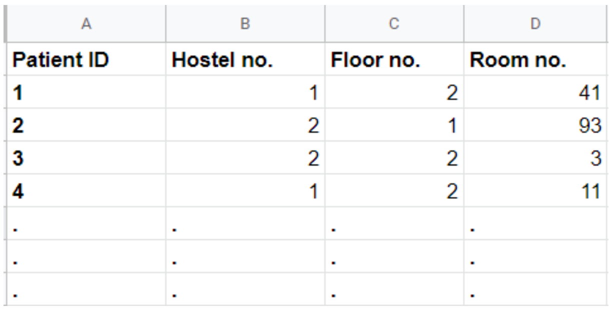
4.1.1 Relational Table-1



Patient ID	Name	Age	Aadhar No.	DoB	Arrival Date	Discharge Date	Coming from	Going to	Phone no.	Relative Contac	House No.	Street Name	Area	City	Pincode	State	Country
1	Shiv Tej	23	11112222	18/05/1998	16/06/2020	30/06/2020	Goa	Silchar	7981313011	Swagat	12	Nehru Street	Secunderabad	Hyderabad	500036	Telangana	India
									9845162981								
2	Abhishek	55	111122223434	14/02/1992	11/07/2020	25/07/2020	Guwahati	Silchar	8162437197	Adarsh	67	Paltan Bazaar	Near Guwahati Station	Guwahati	785203	Assam	India
									9376431691								
									7432761296								
3	Samantha	26	123456781357	17/08/2001	16/07/2020	30/07/2020	Kolkata	Silchar	9656432510	Anamitra	11	Krishna Gali	Jorhat	Jorhat	214562	Assam	India
4	Randeep	32	123443219090	21/12/1999	25/08/2020	08/09/2020	Bangladesh	Silchar	8414262535	Pocham	72	Yashwant Nagar	Jalna	Jalna	431203	Maharashtra	India
.	.	.	.	.	.	.	.	.	9841296321		.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.		.	.	.	.	.	.	.
.	.	.	.	.	.	.	.	.	.		.	.	.	.	.	.	.

In order to establish a relation between the patient and hostel (to improve the efficiency of data access) we add the Patient-id as a foreign key to the hostel entity.

4.1.2 Relational Table-2



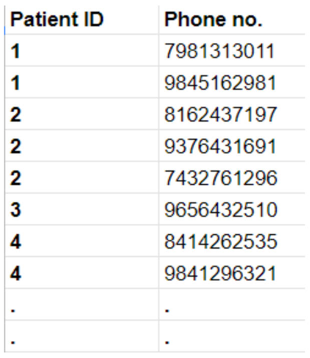
A	B	C	D
Patient ID	Hostel no.	Floor no.	Room no.
1	1	2	41
2	2	1	93
3	2	2	3
4	1	2	11
.	.	.	.
.	.	.	.
.	.	.	.

4.2 Normalization Analysis

4.2.1 1-NF

In First Normal form all the attributes in a relation must have atomic domains. The values in an atomic domain are indivisible units .i.e. It must hold only single-valued attribute. 1NF disallows the multi-valued attribute, composite attribute, and their combinations. Here, phone number is a multi-valued attribute, it has to be stored in a different table. The table for phone number will be created as follows.

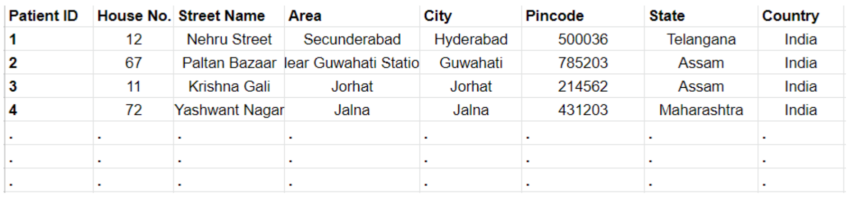
Relational Table-3



Patient ID	Phone no.
1	7981313011
1	9845162981
2	8162437197
2	9376431691
2	7432761296
3	9656432510
4	8414262535
4	9841296321
.	.
.	.

Also Address of a patient is a composite attribute consisting of Street, Area, City, State, PIN code and Country

Relational Table-4



Patient ID	House No.	Street Name	Area	City	Pincode	State	Country
1	12	Nehru Street	Secunderabad	Hyderabad	500036	Telangana	India
2	67	Paltan Bazaar	lear Guwahati Statio	Guwahati	785203	Assam	India
3	11	Krishna Gali	Jorhat	Jorhat	214562	Assam	India
4	72	Yashwant Nagar	Jalna	Jalna	431203	Maharashtra	India
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.
.	.	.	.	.	.	.	.

4.2.2 2-NF

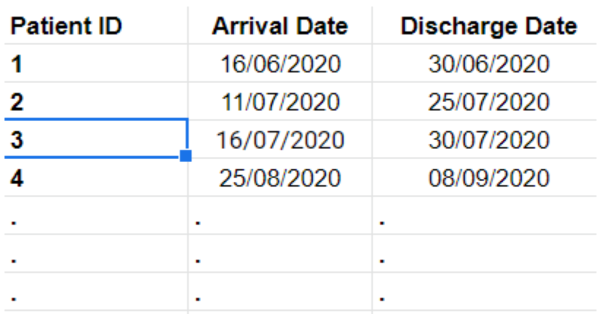
A relation will be in 2NF if it is in 1NF and all non-key attributes are fully functional dependent on the primary key I.e., there must not be any partial dependency. Here no need to change the tables as they are in 1NF and No partial dependency observed.

4.2.3 3-NF

A relation will be in 3NF if it is in 2NF and no transition dependency exists. A transitive dependency in a database is an indirect relationship between values in the same table that causes a functional dependency.

Now, in our relational table Patient, we have a transitive dependency, i.e., Arrival date → Discharge date though neither of them is prime attributes, we can find out the discharge date only in reference to the arrival date which makes it a transitive dependency. To remove the transitive dependency, we have to create another table that contains Arrival Date, Discharge date and the Patient-id and primary key for the table will be P-id.

Relational Table-5



Patient ID	Arrival Date	Discharge Date
1	16/06/2020	30/06/2020
2	11/07/2020	25/07/2020
3	16/07/2020	30/07/2020
4	25/08/2020	08/09/2020
.	.	.
.	.	.
.	.	.

Chapter 5

Triggers and Stored Procedures

A trigger is a stored procedure that gets automatically executed in response to a certain event that takes place in the database. In this code we have created to two triggers :

1. This Trigger automatically adds two weeks to give the discharge date.

```
DROP TRIGGER IF EXISTS ins adm period
CREATE TRIGGER ins adm period
AFTER INSERT ON PATIENT_INFO
FOR EACH ROW
     BEGIN INSERT INTO ADM_PERIOD(Pat_Id,
          Adm_Date, Dis Date)
VALUES (NEW.Pat_ID, CURDATE(),
DATE_ADD(CURDATE(), INTERVAL 14 DAY));
END;
```

2 .We have used a trigger to ensure that all the text entered in the database is correctly formatted, that is for every word the first letter is capital and the rest of the letters are small.

```
DROP TRIGGER IF EXISTS cap on insert
CREATE TRIGGER cap on insert BEFORE
INSERT ON PATIENTINFO
```

```
FOR EACH ROW BEGIN SET NEW.Name =
    camelcase(NEW.Name);
END;


DROP TRIGGER IF EXISTS cap_address
CREATE TRIGGER cap_address BEFORE
INSERT ON PATIENT_ADDRESS
FOR EACH ROW BEGIN SET NEW.Country =
    camelcase(NEW.Country), NEW.State =
    camelcase(NEW.State),
    NEW.city = camelcase(NEW.city),
    NEW.area = camelcase(NEW.area),
    NEW.street = camelcase(NEW.street);
END;
```

Chapter 6

Code

Let’s begin writing the code in steps . Initially As we are using python for creating the user interface , So we first have to connect python to our MySQL for which the code is as follows:

```
import mysql.connector
from datetime import datetime, timedelta
from random import choice
mydb = mysql.connector.connect(
    host = "localhost", user = "root", passwd = "ADARSH1"
)
cur = mydb.cursor()
```

Now the next step is to create a database for the storing and managing data from the normalized tables that we have got from above . Then we go on to create all the above fore mentioned tables .

```
    def create_Database():

qry = "CREATE DATABASE IF NOT EXISTS dbms_project"
cur.execute(qry)

qry = "USE dbms_project"
cur.execute(qry)

qry = "CREATE TABLE IF NOT EXISTS PATIENT_INFO ( " \
  "Pat_ID INT PRIMARY KEY NOT NULL auto_increment, " \
  "Name VARCHAR(255), " \
  "Age INT, " \
  "Arrived_From VARCHAR(255), " \
  "Leaving_To VARCHAR(255)) "
cur.execute(qry)

qry = "CREATE TABLE IF NOT EXISTS ADM_PERIOD ( " \
  "Pat_ID INT REFERENCES PATIENT_INFO(Pat_ID) ON DELETE CASCADE, " \
  "Adm_Date DATE, " \
  "Dis_Date DATE) "
cur.execute(qry)

qry = "CREATE TABLE IF NOT EXISTS PATIENT_ADDRESS ( " \
  "Pat_ID INT REFERENCES PATIENT_INFO(Pat_ID) ON DELETE CASCADE, " \
  "House_Num VARCHAR(255), " \
  "Street VARCHAR(255), " \
  "Area VARCHAR(255), " \
  "City VARCHAR(255), " \
  "Pincode INT, " \
  "State VARCHAR(255), " \
  "Country VARCHAR(255)) "
cur.execute(qry)

qry = "CREATE TABLE IF NOT EXISTS PATIENT_CONTACT ( " \
  "Pat_ID INT REFERENCES PATIENT_INFO(Pat_ID) ON DELETE CASCADE, " \
  "Phone_Num VARCHAR(255)) "
cur.execute(qry)

qry = "CREATE TABLE IF NOT EXISTS ROOM_INFO ( " \
       "Pat_ID INT REFERENCES PATIENT_INFO(Pat_ID) ON DELETE CASCADE, "
       "Hostel_Num INT, " \
       "Floor_Num INT, " \
       "Room_Num INT) "
cur.execute(qry)

mydb.commit()
```

Now every time we have to insert a record in a table , we are going to call insert_Record() and it will insert data into all the table using MySQL commands .

```
    def insert_Record():

# Insert Patient Info , Phone Numbers , Patient Address , Room Info
```

```
qry = "INSERT INTO PATIENT_INFO (NAME, Age, Arrived_From, Leaving_To) "
"VALUES (%s, %s, %s, %s) "

name = input("Enter Patient Name : ")
age = int(input("Enter Age : "))
arr_frm = input("From where did  you  come?  ")
lev_to = input("Where will you be heading to? ")

cur.execute(qry, (name, age, arr_frm, lev_to))

mydb.commit()

qry = "INSERT INTO PATIENT_CONTACT (Pat_ID, Phone_Num) " \
   " VALUES (%s, %s) "
phone_num = [int(x) for x in input("Enter Phone Numbers
separated by space :  ").split()]
data = []

getqry = "SELECT  LAST_INSERT_ID()  "
cur.execute(getqry)
pat_id = 0

for x in cur:
for id in x:
pat_id = id

for num in phone_num:
data.append((pat_id,   num))

cur.executemany(qry, data)

qry = "INSERT INTO PATIENT_ADDRESS (Pat_ID, House_Num,
Street, Area, City, Pincode, State, Country) " \
        "VALUES (%s, %s, %s, %s, %s, %s, %s, %s) "

house_num = input("Enter House Number : ")
street = input("Enter  Street  Name  : ")
area = input("Enter Area Name : ")
city = input("Enter City Name : ")
pincode = int(input("Enter Area’s Pincode : "))
state = input("Enter State : ")
```

```
country = input("Enter Country : ")

cur.execute(qry, (pat_id, house_num, street, area, city,
pincode,  state,  country))


qry = "INSERT INTO ROOM_INFO (Pat_ID, Hostel_Num, Floor_
Num, Room_Num) " \
   "VALUES  (%s,  %s,  %s,  %s)  "

room_num = 0;

if  age >= 60:
room_num  =  choice(room[0])
room[0].remove(room_num)
elif age >= 40:
room_num  =  choice(room[1])
room[1].remove(room_num)
else:
room_num = choice(room[2])
room[2].remove(room_num)

floor_num = (room_num // 100) % 10
hostel_num = room_num // 1000
room_num %= 100

cur.execute(qry,  (pat_id,  hostel_num,  floor_num,  room_num))

mydb.commit()

# Display alloted room
print("Assigned Hostel " +
str(hostel_num) + " Floor " +
str(floor_num) + " Room " + str(room_num) +
" to Patient " + str(name) + " (ID : " +
str(pat_id) + ")")
print("")
```

Now we also want to extract information about any person quarantined in the hostel at any moment . For that purpose what we are trying to do here is to find records of the person using his allotted patient Id and returning the information of the person in a map/dictionary mp .

```
      def  getData(id):

mp =  {}

qry = "SELECT  *  FROM  PATIENT_INFO  WHERE  Pat_ID  =  %s  "
cur.execute(qry, (id, ))

for  x  in  cur:
mp["pat_id"] = x[0]
mp["name"]  =  x[1]
mp["age"]  =  x[2]
mp["com_frm"]  =  x[3]
mp["lev_to"] = x[4]

qry  =  "SELECT  *  FROM  ADM_PERIOD  WHERE  Pat_ID  =  %s  "
cur.execute(qry, (id, ))

for  x  in  cur:
mp["adm_date"]  =  x[1]
mp["dis_date"]  =  x[2]

qry = "SELECT  Phone_Num  FROM  PATIENT_CONTACT  WHERE  Pat_ID  =  %s  "
cur.execute(qry, (id, ))

mp["phn"]  =  []
for x in cur:
for  num  in  x:
mp["phn"].append(num)

qry =  "SELECT * FROM  PATIENT_ADDRESS  WHERE  Pat_ID  =  %s  "
cur.execute(qry, (id, ))

for x in cur:
mp["house_num"] = x[1]
mp["street"] = x[2]
mp["area"]  =  x[3]
mp["city"]  =  x[4]
```

```
mp["pincode"] = x[5]
mp["state"] = x[6]
mp["country"] = x[7]

qry = "SELECT * FROM ROOM_INFO WHERE Pat_id = %s "
cur.execute(qry, (id, ))

for x in cur:
mp["hostel_num"] = x[1]
mp["floor_num"] = x[2]
mp["room_num"] = x[3]

return mp
```

Now in order to display the data for a given patient using the map/dictionary mp :

```
    def showData(mp):

# This function prints the data using the dictionary
print("")
print("Patient Id : " + str(mp["pat_id"]))
print("Name : " + str(mp["name"]))
print("Age :  " + str(mp["age"]))
print("Coming From : " + str(mp["com_frm"]))
print("Leaving To :  " + str(mp["lev_to"]))
print("Admission Date : " + str(mp["adm_date"]))
print("Discharge Date : " + str(mp["dis_date"]))
print("Address : " +
str(mp["house_num"]) + "  " + str(mp["street"]) +
"  " + str(mp["area"]) + ", " + str(mp["city"]) +
", " + str(mp["state"]) + ", " + str(mp["country"]))
print("PinCode : " + str(mp["pincode"]))
print("Phone Numbers :", *mp["phn"])
print("Hostel : " + str(mp["hostel_num"]) +
", Floor : " + str(mp["floor_num"]) +
", Room : " + str(mp["room_num"]))
print("")
print("-------------
-------------------------------------------------------------
-------------")
```

Now to view records of a person initially , we have to either know allotted person’s ID or we need to know the name .

```
def view_Records():

qry = "SELECT Pat_ID from PATIENT_INFO "
cur.execute(qry)

pat_id = []
for x in cur:
for id in x:
pat_id.append(id)

pat_id.sort()

if len(pat_id) == 0:
print("No records found")
else:
data = []

for id in pat_id:
mp = getData(id)
data.append(mp)

for mp in data:
showData(mp)
print("")


def search_Record():

# This function is to print the data of a particular patient, either
# by id or by name. It then uses getData and showData functions to
# get and print the data
c = int(input("Enter
\n1 to search using Patient ID\n2 to Search using Name : "))
pat_id = []
if(c == 1):
x = int(input("Enter Patient ID : "))
```

```
getqry = "SELECT  Pat_ID  FROM  PATIENT_INFO  WHERE  Pat_ID  =  %s  "
cur.execute(getqry, (x, ))

for x in cur:
pat_id.append(int(x[0]))

else:
name = input("Enter Patient Name : ")

getqry =  "SELECT  Pat_ID  FROM  PATIENT_INFO WHERE  Name  =  %s  "
cur.execute(getqry,  (name,  ))

for x in cur:
for id in x:
pat_id.append(int(id))

data = []
for id in pat_id:
mp = getData(id)
data.append(mp)

if len(data) == 0:
print("No records found")
else:
for mp in  data:
showData(mp)
print("")
```

Now we might need to delete some records which might be redundant after the person leaves the premise , for that purpose delete_Record() is being called , which deletes the record from all the tables .

```
    def delete_Record():

# Delete record from all 5 tables where Pat_ID = id
# Add the room number of discharged patient back to vacant rooms
pat_id = input("Enter the Patient ID :  ")

qry = "SELECT Hostel_num, Floor_Num, Room_Num from ROOM_INFO "
cur.execute(qry)
```

```
room_num, floor_num, hostel_num = 0, 0, 0

for x in cur:
room_num = x[0]
floor_num  =  x[1]
room_num = x[2]

if room_num == 0:
return

room_num = int(hostel_num) * 1000 +
int(floor_num)  *  100 + int(room_num)

room[floor_num].append(room_num)

qry = "DELETE  FROM  PATIENT_INFO  WHERE  Pat_ID  =  %s  "
cur.execute(qry, (pat_id, ))

qry = "DELETE FROM PATIENT_ADDRESS WHERE Pat_ID = %s "
cur.execute(qry, (pat_id, ))

qry = "DELETE FROM PATIENT_CONTACT WHERE Pat_ID = %s "
cur.execute(qry, (pat_id, ))

qry = "DELETE  FROM  ROOM_INFO  WHERE  Pat_ID  =  %s  "
cur.execute(qry, (pat_id, ))

qry = "DELETE FROM ADM_PERIOD WHERE Pat_id = %s "
cur.execute(qry, (pat_id, ))

mydb.commit()

print("Record deleted successfully")
print("")
```

This function calls all the triggers to start working .

```
def use_triggers():

# This function capitalizes the first  letter  of  the  name .
qry = "DROP FUNCTION IF EXISTS camelcase "
```

```
cur.execute(qry)
```

This function takes the string and scans it from left to right As it encounters a puncutation, it capitalizes the next character

```
    qry = "CREATE FUNCTION ‘camelcase‘( str VARCHAR(255) )
    RETURNS VARCHAR(255) CHARSET utf8 DETERMINISTIC BEGIN
    DECLARE c CHAR(1); DECLARE s VARCHAR(255);
    DECLARE i INT DEFAULT 1;
    DECLARE bool INT DEFAULT 1;
    DECLARE punct CHAR(17)
    DEFAULT ’ ()[]{},.-_!@;:?/’;
    SET s = LCASE( str );
    WHILE i < LENGTH( str )
    DO BEGIN      SET c = SUBSTRING( s, i, 1 );
    IF LOCATE( c, punct ) > 0
    THEN     SET bool = 1;
    ELSEIF bool=1 THEN
    BEGIN
    IF c >= ’a’ AND c <= ’z’
    THEN     BEGIN    SET s = CONCAT(LEFT(s,i-1),
    UCASE(c),
    SUBSTRING(s,i+1));
    SET bool = 0;
    END;
    ELSEIF c >= ’0’ AND c <= ’9’
    THEN SET bool = 0;
    END IF; END;
    END IF;
    SET i = i+1; END;
    END WHILE;
    RETURN s;
    END"
      cur.execute(qry)

mydb.commit()

    # Creating a trigger to convert name to camel case
    when inserting a record
    qry = "DROP TRIGGER IF EXISTS cap_on_insert "
    cur.execute(qry)
```

```
qry = "CREATE TRIGGER cap_on_insert
BEFORE INSERT ON PATIENT_INFO
FOR EACH ROW BEGIN
SET NEW.Name = camelcase(NEW.Name);
END; "
cur.execute(qry)


# Creating a trigger to convert country,
state, city, area and street
to camel case when inserting a record
qry = "DROP TRIGGER IF EXISTS cap_address "
cur.execute(qry)

qry = "CREATE TRIGGER cap_address
BEFORE INSERT ON PATIENT_ADDRESS
FOR EACH ROW
BEGIN
SET NEW.Country = camelcase(NEW.Country),
NEW.State = camelcase(NEW.State),
NEW.city = camelcase(NEW.city),
NEW.area = camelcase(NEW.area),
NEW.street = camelcase(NEW.street);
END; "
cur.execute(qry)

# Inserting admission date and discharge
date to admission period table
# as a new record is inserted into
patient details table
qry = "DROP TRIGGER IF EXISTS ins_adm_period "
cur.execute(qry)

qry = "CREATE TRIGGER ins_adm_period
AFTER INSERT ON PATIENT_INFO
FOR EACH ROW
BEGIN
INSERT INTO ADM_PERIOD(Pat_Id,
Adm_Date, Dis_Date)
VALUES (NEW.Pat_ID, CURDATE(),
DATE_ADD(CURDATE(),
INTERVAL 14 DAy));
END; "
cur.execute(qry)

mydb.commit()
```

Allocation for rooms is being done by 2D array room with appending rooms which can be occupied .

```
    room = [[] for _ in  range(3)]

def roomAllocation():

        for r in range(1000, 1100):
                room[0].append(r)
        for r in range(2000, 2100):
                room[0].append(r)

        for r in range(1100, 1200):
                room[1].append(r)
        for r in range(2100, 2200):
                room[1].append(r)

        for r in range(1200, 1250):
                room[2].append(r)
        for r in range(2200, 2250):
                room[2].append(r)
```

Calling all the above functions for creating database , allocating rooms and making triggers start working .

```
    roomAllocation()

    create_Database()

    use_triggers()
```

This piece of code is for removal of rooms which are being still occupied by the people residing in the hostel rooms .

```
qry = "SELECT Hostel_Num, Floor_Num,
Room_Num from ROOM_INFO "
cur.execute(qry)
for x in cur:
    room[x[1]].remove(int(x[0])  *  1000
    + int(x[1]) *  100
    +  int(x[2]))
```

Now this piece of code gives a user interface with options to pick when the code runs .

```
    while True:

c = int(input("Enter \n1 to Insert a Patient’s Details
\n2 to View all Records
\n3 to Search a Patient
\n4 to Delete a Patient’s Details
\n5 to Exit :
"))

if c == 1:
insert_Record()
elif c == 2:
view_Records()
elif c == 3:
search_Record()
elif c == 4:
delete_Record()
else:
break
```

Closing connection to end the program .

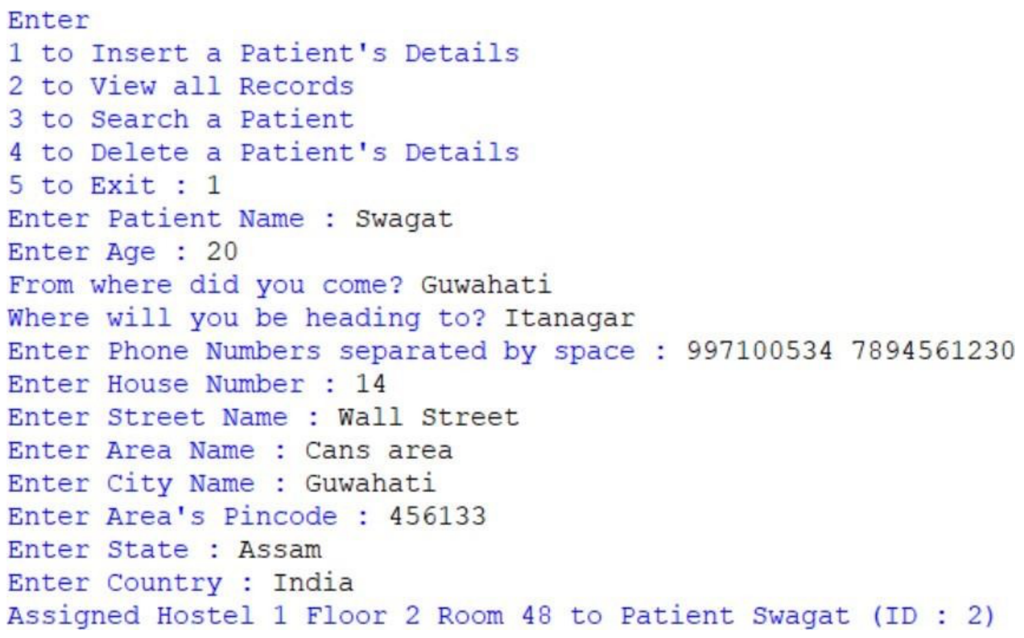
```
cur.close()
mydb.close()
```

Chapter 7

Code Execution

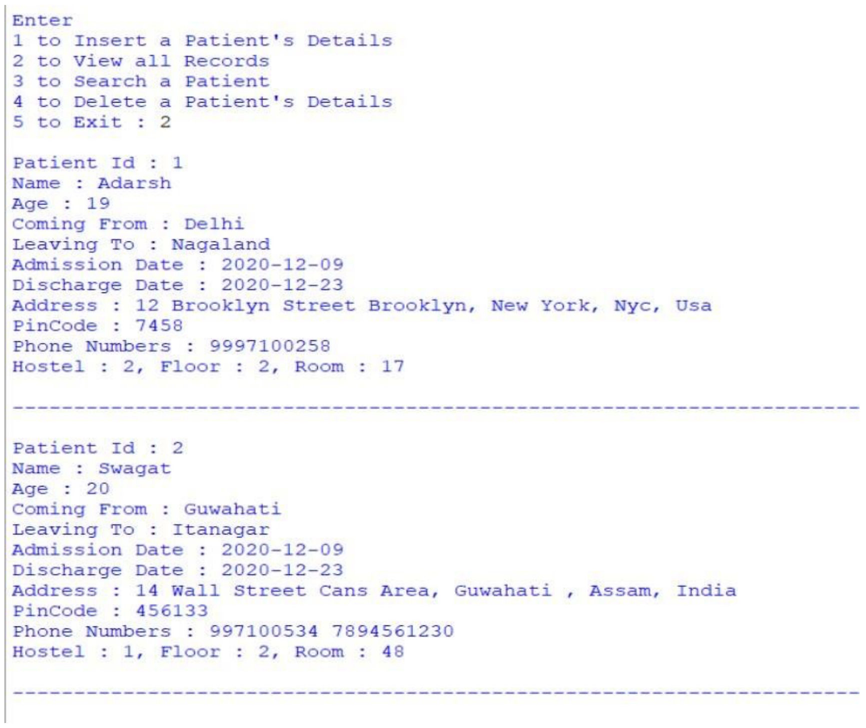
***NOTE:** Data entered below should not be confused with dummy values put in previous tables, this is purely for exemplary purposes....*

7.1 Inserting a record



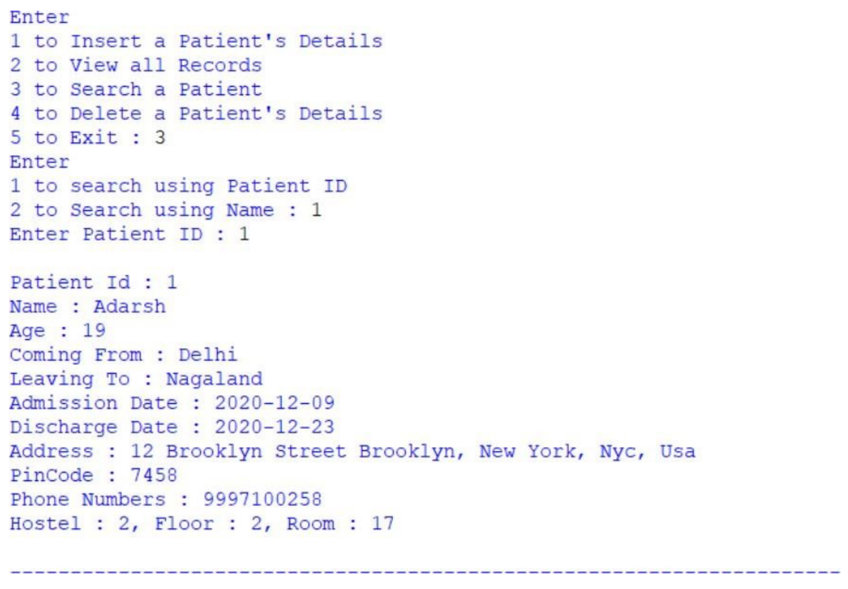
```
Enter
1 to Insert a Patient's Details
2 to View all Records
3 to Search a Patient
4 to Delete a Patient's Details
5 to Exit : 1
Enter Patient Name : Swagat
Enter Age : 20
From where did you come? Guwahati
Where will you be heading to? Itanagar
Enter Phone Numbers separated by space : 997100534 7894561230
Enter House Number : 14
Enter Street Name : Wall Street
Enter Area Name : Cans area
Enter City Name : Guwahati
Enter Area's Pincode : 456133
Enter State : Assam
Enter Country : India
Assigned Hostel 1 Floor 2 Room 48 to Patient Swagat (ID : 2)
```

7.2 View all records



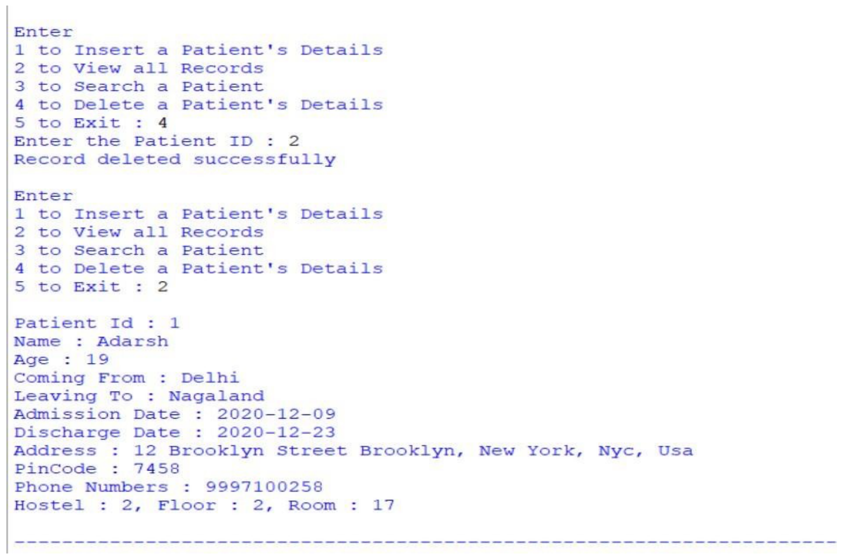
```
Enter
1 to Insert a Patient's Details
2 to View all Records
3 to Search a Patient
4 to Delete a Patient's Details
5 to Exit : 2

Patient Id : 1
Name : Adarsh
Age : 19
Coming From : Delhi
Leaving To : Nagaland
Admission Date : 2020-12-09
Discharge Date : 2020-12-23
Address : 12 Brooklyn Street Brooklyn, New York, Nyc, Usa
PinCode : 7458
Phone Numbers : 9997100258
Hostel : 2, Floor : 2, Room : 17

----------------------------------------------------------------------

Patient Id : 2
Name : Swagat
Age : 20
Coming From : Guwahati
Leaving To : Itanagar
Admission Date : 2020-12-09
Discharge Date : 2020-12-23
Address : 14 Wall Street Cans Area, Guwahati , Assam, India
PinCode : 456133
Phone Numbers : 997100534 7894561230
Hostel : 1, Floor : 2, Room : 48

----------------------------------------------------------------------
```

7.3 Search a record



```
Enter
1 to Insert a Patient's Details
2 to View all Records
3 to Search a Patient
4 to Delete a Patient's Details
5 to Exit : 3
Enter
1 to search using Patient ID
2 to Search using Name : 1
Enter Patient ID : 1

Patient Id : 1
Name : Adarsh
Age : 19
Coming From : Delhi
Leaving To : Nagaland
Admission Date : 2020-12-09
Discharge Date : 2020-12-23
Address : 12 Brooklyn Street Brooklyn, New York, Nyc, Usa
PinCode : 7458
Phone Numbers : 9997100258
Hostel : 2, Floor : 2, Room : 17

----------------------------------------------------------------------
```

7.4 Delete a record



```
Enter
1 to Insert a Patient's Details
2 to View all Records
3 to Search a Patient
4 to Delete a Patient's Details
5 to Exit : 4
Enter the Patient ID : 2
Record deleted successfully

Enter
1 to Insert a Patient's Details
2 to View all Records
3 to Search a Patient
4 to Delete a Patient's Details
5 to Exit : 2

Patient Id : 1
Name : Adarsh
Age : 19
Coming From : Delhi
Leaving To : Nagaland
Admission Date : 2020-12-09
Discharge Date : 2020-12-23
Address : 12 Brooklyn Street Brooklyn, New York, Nyc, Usa
PinCode : 7458
Phone Numbers : 9997100258
Hostel : 2, Floor : 2, Room : 17

---------------------------------------------------------------------
```

Chapter 8

Conclusion

With the current situation of the pandemic menacing the world, it is of utmost importance for associations to come up with efficient Quarantine System Management tools using Databases, now more than ever. Although many hospitals and Quarantine Centers have been using a database for ages, maintaining an efficient DBMS system is still a very important task for any organization. We would hence be implementing such a Patient Record maintenance system for a Quarantine Facility, using a DBMS system that is very efficient in handling such large patient record datasets.

In conclusion, a database is a far more efficient mechanism to store and organize data than spreadsheets. For a centralized facility it facilitates easy and quick modification of data shared among multiple users. With this mini project, we demonstrated how easily patient records can now be stored and accessed with simple CRUD operations (Create, Read, Update, Delete), making customer service a priority with enhanced and easy updating and discharging of patients, in the given Quarantine Facility. It also allows the possibility of queries to obtain information for various patients. With the number of patients changing constantly it is elective and an ideal use for this a system.