**IS 620 Advanced Database Projects**

Group Project D2: **Design Document**

Group Members: Kirubel Tolosa, Sonal Ingle, Shubhi Shrivastava, Vikas Matadha

**SQL statements to create database tables:**

**drop table** Person\_flight cascade constraints;

**drop table** Flights cascade constraints;

**drop table** Person\_event cascade constraints;

**drop table** Events cascade constraints;

**drop table** Tests cascade constraints;

**drop table** Person cascade constraints;

**drop table** Houses cascade constraints;

**CREATE TABLE** Houses

(

hid INT,

address VARCHAR(30),

zip CHAR(5),

PRIMARY KEY(hid)

);

**CREATE TABLE** Person

(

pid INT,

hid INT,

p\_name VARCHAR(15),

phone VARCHAR(10),

status BIT,

PRIMARY KEY(pid)

FOREIGN KEY(hid) REFERENCES Houses(hid)

);

**CREATE TABLE** Tests

(

pid INT,

t\_date DATE,

result BIT,

FOREIGN KEY(pid) REFERENCES Person(pid)

);

**CREATE TABLE** Events

(

eid INT,

e\_name VARCHAR(20),

e\_date DATE,

e\_address VARCHAR(30),

PRIMARY KEY(eid)

);

**CREATE TABLE** Person\_event

(

pid INT,

eid INT,

FOREIGN KEY(pid) REFERENCES Person(pid)

FOREIGN KEY(eid) REFERENCES Events(eid)

);

**CREATE TABLE** Flights

(

fid INT,

f\_date DATE,

f\_number VARCHAR(10),

PRIMARY KEY(fid)

);

**CREATE TABLE** Person\_flight

(

pid INT,

fid INT,

FOREIGN KEY(pid) REFERENCES Person(pid),

FOREIGN KEY(fid) REFERENCES Flights(fid)

);

**Table Inserts:**

insert values into table Houses

insert into Houses values(1, '1025 Howland Sq', ‘21227’);

insert into Houses values(2, '1026 Howland Sq', ‘21228’);

insert into Houses values(3, '1027 Howland Sq', ‘21229’);

insert into Houses values(4, '1028 Howland Sq', ‘21220’);

insert into Houses values(5, '1029 Howland Sq', ‘21221’);

insert values into table Person

insert into Person values(11, 1, ’112 Aldgate MD’, ‘Rohan’, ’6672321002’, 0);

insert into Person values(12, 2, '113 Aldgate MD’, ‘Vikas’, ‘6672321003’, 0);

insert into Person values(13, 3, '114 Aldgate MD’, ‘Vickey’, ’6672321004’, 1);

insert into Person values(14, 4, '115 Aldgate MD’, ‘Vaishak’, ‘6672321005’, 1);

insert into Person values(15, 5, '116 Aldgate MD’, ‘Vishal’, ‘6672321006’, 1);l

insert values into table Tests

insert into Tests values(11, date '2020-08-08', 0);

insert into Tests values(12, date '2020-10-07', 0');

insert into Tests values(13, date '2010-08-08', 1);

insert into Tests values(11, date '2019-08-05', 0);

insert into Tests values(14, date '2020-09-01', 0);

insert values into table Events

insert into Events values(311, 'Victor', date '2011-09-05', '4142 greenville');

insert into Events values(312, 'Hitler', date '2013-10-03', '4143 greenville');

insert into Events values(313, 'Vile', date '2015-08-05', '4144 greenville');

insert into Events values(314, 'John', date '2016-10-02', '4145 greenville');

insert into Events values(315, 'Peter', date '2017-01-05', ‘4146 greenville');

insert values into table Person\_event

insert into Person\_event values(311, 11);

insert into Person\_event values(312, 12);

insert into Person\_event values(313, 13);

insert into Person\_event values(314, 11);

insert into Person\_event values(315, 14);

insert values into table Flights

insert into Flights values(1111, date '2020-06-13', 819);

insert into Flights values(1112, date '2020-10-08', 820);

insert into Flights values(1113, date '2017-08-18', 821);

insert into Flights values(1114, date '2020-10-28', 823);

insert into Flights values(1115, date '2010-11-08', 824);

insert values into table Person\_flight

insert into Person\_flight values(11, 1111);

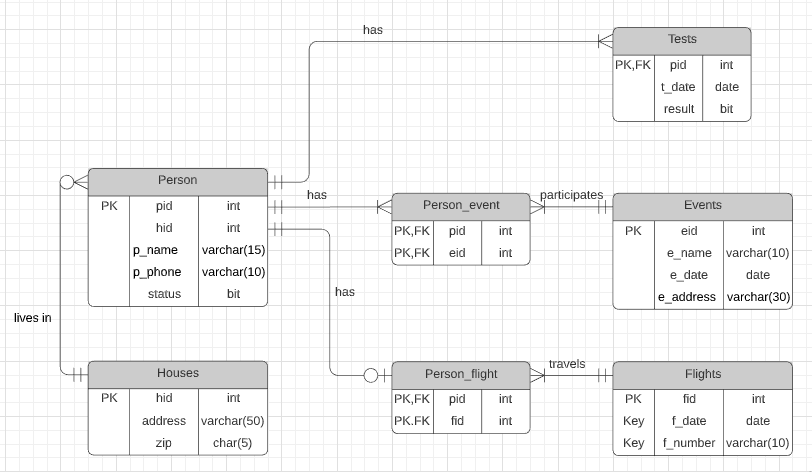
insert into Person\_flight values(12, 1112);

insert into Person\_flight values(11, 1111);

insert into Person\_flight values(13, 1114);

insert into Person\_flight values(14, 1111);

**Entity Relationship Diagram:**



**# Feature 1:**

Input: name of person, home address and zip code, phone\_number,status=NULL.

Output: print a message in different cases:

1) Person already exists;

2) Newly assigned person ID with existing house;

3) New assigned person ID and newly generated house ID.

Calling the procedure:

exec add\_person('Sonal','123 Belwood Green','21227','1-443-987-1028'); --- add\_person is name of procedure

**# Feature 2:**

Input: Set the status = 1.

Output: print the message if the status =1:

1) P Person name, phone\_number, status;

2) Person name, phone\_number, status of all members whose house id is same as of person P excluding P.

Calling the procedure:

exec list\_person(status); --- [list\_person is the name of the procedure]

sample o/p : ('Bob','1-443-999-1010',1)

**# Feature 3:**

Input: person id, test result and test date

Output: print in different cases:

1) error message: person with the entered id does not exist

2) newly assigned or updated test result and test date for person with entered pid

Calling the procedure : exec add\_test(person id, test date, test result) --- [add\_test is the name of the procedure]

**# Feature 4:**

Input: X, which is number of days/time range

Output: list of P with status = 1 and with same flight

1. P Person name and phone number
2. people who were on the same flight as them in the last X days

Calling the procedure : exec list\_people(X) --- [list\_people is the name of the procedure]

**# Feature 5:**

Enter a new event.

Input: event date, name, and address.

Output:

Case 1: To first check whether an event with the same name, date, and address exits. If the event exists, then print a message event that exists.

Case 2: If the event doesn’t exist then, insert the row in the event table with event date, name and address and print the event id.

Calling the procedure: exec add event(311, 'Victor', date '2011-09-05', '4142 greenville');

**# Feature 6:**

Input: status = 1

Output: People name and phone number whose status = 1 and run a loop for each person and return names and phone number of person who attended the same event as person with status = 1 attended in last x days.

Calling the procedure:

Exec check\_PositivePerson\_SameEvent(person\_name, person\_phone);

**# Feature 7:**

**Inputs:**

P1: plist **personsListType**

P2: pid **integer**

**Outputs:**

Event does not exist -- (When no event with id is found in the Event table)

Person does not exist -- (When no person with the id is recorded in the Person table)

No need to insert -- (When a match is already registered in the Person\_Event table)

Registered Person: 1 at Event: 3 -- (When Person\_Event table is updated)

**Usage:**

exec insertPersonsAtEvent(personsListType(1, 3, 6), 2);

**# Feature 8:**

**Inputs:**

P1: p\_name **varchar(15)**

P2: Phone **Number**

**Outputs:**

No such person! -- (when no person matching the name and phone number exists)

No test records for the person -- (when no records found for the person in the Tests table)

-- or the following

10-13-2020 : Negative

10-03-2020 : Negative

09-22-2020 : Negative

**Usage:**

exec printTests('somename', '123-456-7890');

**# Feature 9:**

Input: person id, status =1, date D, X, Y ---[X,Y are number of days]

Output: print a message in different cases:

1) Person A exists, then information - person id, date;

2) Person id and name having same house id as that of person A excluding person A;

3) Person id and name having same event id as that of person A where the event date is between (date D - X) and (date D) excluding person A;

4) Person id and name having same flight id as of person A where event date is between (date D - X) and (date D) excluding person A;

5) Person name B in close contact with person A whose test result is positive between D + 1 and D + Y;

6) Reason for person B [choose from point 2,3,4];

7) Unique person id and person name from list in point 5;

Calling the procedure:

exec person\_check(person id, status, date D); --- [person\_check is the procedure name]

exec recursive\_close\_contact(person id, status, date D, X, Y); --- [recursive\_close\_contact is name a procedure]

# **Feature 10:**

1. Input:none

Output: total count of people who have status=1 and count of such people in each zip code as groups

Calling the procedure : exec number\_Positives()

--- [number\_Positives is the name of the procedure]

1. Input:X, which is number of days

Output: total count of distinct people who have tested positive in last X days

Calling the procedure: exec number\_Positives\_X(X)

--- [number\_Positives\_X is the name of the procedure]

1. Input:none

Output: accumulated count of distinct people who have tested positive at least once in each zip code(group by zip code)

Calling the procedure: exec number\_Positives\_ZipCode()

--- [number\_Positives\_ZipCode is the name of the procedure]

**# Feature 11:**

Daily stats. Given a start date and end date, print out daily statistics in this period.

Input: start date, end date

Output:

Case 1: Prints out the daily accumulated positive cases between the input start data and input end date.

-prints out daily new cases. Includes people were not positive until that date.

-prints out daily current cases. Includes people who tested positive on or before the current date.

-prints out daily accumulated recovered cases. Includes people who tested negative on or before that date, but tested positive before.

Case 2: If the dates don’t return any results, print ‘no data for these dates.’

Calling the procedure:

Execute example: exec daily\_stat(date ‘2020-04-05’, date ‘2020-06-06’);

**# Feature 12:**

**Inputs:**

P1: hotSpot\_date **DATE**

P2: days **Number**

P3: threshold  **Number**

**Outputs:**

Hot spot Zip Codes:

12345

23456

..

**Usage:**

exec findHotSpots( Date '10-13-2020', 25, 1200);