

Practicum-1 Design and Implementation of Relational Database - Contact Tracer

Varshitha Uppula
Saisrihitha Yadlapalli
(Group 3)

1.Introduction

Contact Tracing, a core disease control measure employed by local and state health department personnel for decades, is a key strategy for preventing further spread of COVID-19. A relational database has been implemented to store information about a mobile application user who is tracked based on GPS which would aid to find the users who have come in contact with the infected patients. Surveys are conducted for users who can record their data about temperature, symptoms, etc. Additionally, information about labs and tests conducted in particular cities is maintained.

2.Assumptions

The following are the key assumptions that we made with respect to developing the contact tracing database/models:

1. We have modeled our application such that it can be used for contact tracing for the Covid-19 pandemic, but with minor modifications it can be used for generic contact tracing and for analytical use cases as well.
2. An app user is anyone who can use the contact tracing application and can be divided into two categories: a **naïve user** and a **health monitor admin**. An app user can have only a **single phone number**.
3. A naïve user is one who can use the application and is regularly surveyed for symptoms and tracked to check if they have come in contact with an infected user. A naïve user is either **infected**, **not-infected** or **suspected** based on the **infection status** which is given to him/her based on the results of tests conducted by a particular lab that are verified by the health monitor admin.
4. A health monitor admin can view information regarding naïve users and will be notified if a naïve user in a region that the admin is responsible for monitoring has a high risk of having contracted Covid-19.
5. A naïve user will also be notified if he/she is at **risk** of contracting Covid-19. The risk (can be calculated using triggers) of a naïve user having contracted Covid-19 depends upon various factors such as:
 - user symptoms and other information that get recorded as a part of regular surveys
 - medical condition of the user that compromises their immunity
 - travel history of user within a given time period
 - user coming in contact with zero or more infected user based on GPS tracking.
6. A naïve user is tracked based on GPS coordinates and we have considered "gpsLoc" as an attribute that uniquely identifies the coordinates the user on a particular date as we have considered the virus to be active/present where the infected person has been to for at least one day.

3. Conceptual model

Conceptual model showing the various entities at a high level is built using **Visual Paradigm**.

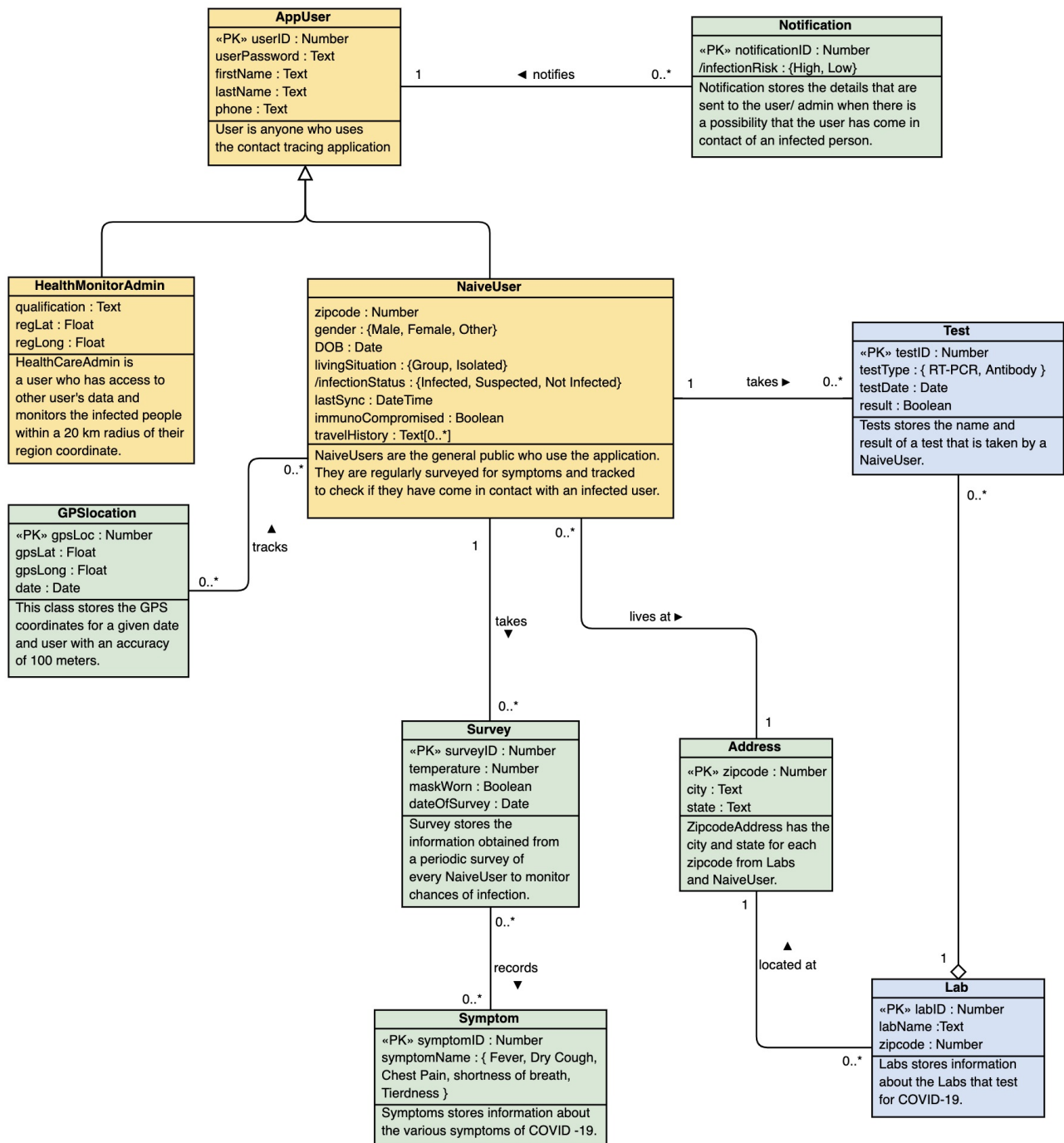


Fig 3.1: Conceptual Model for Contact Tracing in Visual Paradigm

4. Logical Model

Logical diagram with Crow's Foot Notation was built using **Lucid Chart**.

Link to Logical Model (<https://app.lucidchart.com/invitations/accept/55d3e1a0-dc6e-4a04-8d6c-6d30657dec3e>)

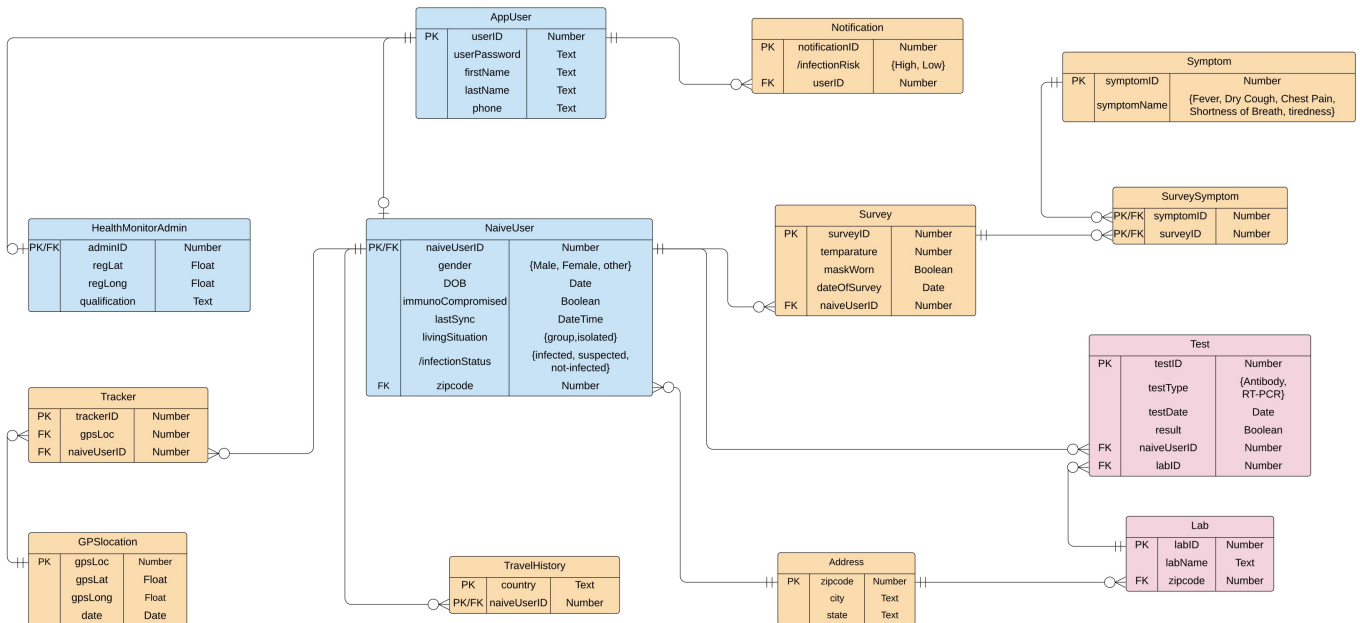


Fig 4.1: Logical Model for Contact Tracing in Lucid Chart

5. Relational Schema Definitions

The logical schema shown above is resolved into the below relational schema. A relation R with attributes $A_1, A_2, A_3 \dots A_n$ is shown as $R(\underline{A_1}, A_2, A_3 \dots A_n)$, where the primary key is underlined and foreign keys are shown in *italics*.

AppUser (userID, userPassword, firstName, lastName, phone)

HealthMonitorAdmin (adminID, regLat, regLong, qualification)

NaiveUser (naiveUserID, gender, immunoCompromised, DOB, lastSync, livingSituation, infectionStatus, zipcode)

Tracker (trackerID, gpsLoc, naiveUserID)

GPSlocation (gpsLoc, gpsLat, gpsLong, date)

TravelHistory (country, *naiveUserID*)

Survey (surveyID, temperature, maskWorn, dateOfSurvey, *naiveUserID*)

Symptom (symptomID, symptomName)

SurveySymptom (symptomID, surveyID)

Test (testID, testType, testDate, result, *naiveUserID*, *labID*)

Lab (labID, labName, zipcode)

Address (zipcode, city, state)

Notification (notificationID, infectionRisk, *userID*)

6. Proof using functional dependencies to show that schema is in BCNF:

A relational schema is considered to be in BCNF if for every one of its dependencies $X \rightarrow Y$ one of the following conditions holds true:

$X \rightarrow Y$ is a trivial functional dependency (i.e Y is a subset of X)

X is a superkey for the schema

AppUser

$userID \rightarrow \{ userPassword, firstName, lastName, phone \}$

$phone \rightarrow \{ userID, userPassword, firstName, lastName \}$

(Here, there are two candidate keys $userID$ and $phone$ which uniquely identify the non-key attributes and hence satisfies BCNF)

HealthMonitorAdmin

$adminID \rightarrow \{ regLat, regLong, qualification \}$

NaiveUser

$naiveUserID \rightarrow \{ gender, immunoCompromised, DOB, lastSync, livingSituation, infectionStatus, zipcode \}$

GPSlocation

$gpsLoc \rightarrow \{ gpsLat, gpsLong, date \}$

Tracker

$trackerID \rightarrow \{ gpsLoc, naiveUserID \}$

TravelHistory

$country, naiveUserID$ form a composite primary key (trivial FD hence in BCNF)

Survey

$surveyID \rightarrow \{ temperature, maskWorn, dateOfSurvey, naiveUserID \}$

Symptom

$symptomID \rightarrow symptomName$

SurveySymptom

$surveyID, symptomID$ form a composite primary key (trivial FD hence in BCNF)

Test

$testID \rightarrow \{ testType, testDate, result, naiveUserID, labID \}$

Lab

$labID \rightarrow \{ labName, zipcode \}$

Address

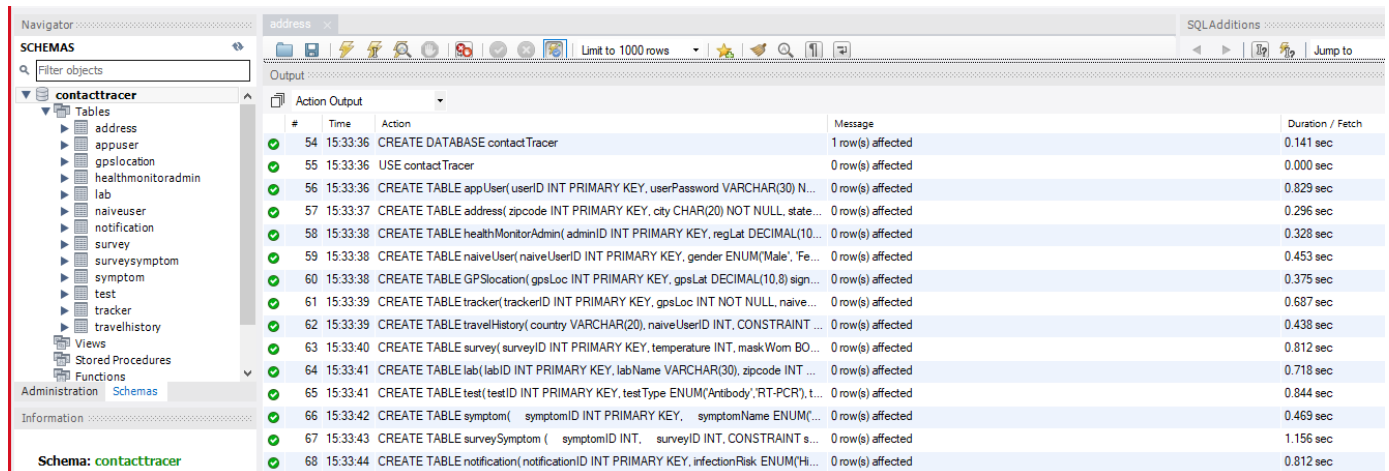
$Zipcode \rightarrow \{ city, state \}$

Notification

notificationID → { infectionRisk, userID }

7. Schema creation and Table definition in MySQL

Tables created in MySQL database using CREATE statements



#	Time	Action	Message	Duration / Fetch
54	15:33:36	CREATE DATABASE contactTracer	1 row(s) affected	0.141 sec
55	15:33:36	USE contactTracer	0 row(s) affected	0.000 sec
56	15:33:36	CREATE TABLE appUser(userID INT PRIMARY KEY, userPassword VARCHAR(30) N...	0 row(s) affected	0.829 sec
57	15:33:37	CREATE TABLE address(zipcode INT PRIMARY KEY, city CHAR(20) NOT NULL, state...	0 row(s) affected	0.296 sec
58	15:33:38	CREATE TABLE healthMonitorAdmin(adminID INT PRIMARY KEY, regLat DECIMAL(10...	0 row(s) affected	0.328 sec
59	15:33:38	CREATE TABLE naiveUser(naiveUserID INT PRIMARY KEY, gender ENUM('Male', 'Fe...	0 row(s) affected	0.453 sec
60	15:33:38	CREATE TABLE GPSlocation(gpsLoc INT PRIMARY KEY, gpsLat DECIMAL(10.8) sign...	0 row(s) affected	0.375 sec
61	15:33:39	CREATE TABLE tracker(trackerID INT PRIMARY KEY, gpsLoc INT NOT NULL, naive...	0 row(s) affected	0.687 sec
62	15:33:39	CREATE TABLE travelHistory(country VARCHAR(20), naiveUserID INT, CONSTRAINT ...	0 row(s) affected	0.438 sec
63	15:33:40	CREATE TABLE survey(surveyID INT PRIMARY KEY, temperature INT, maskWorn BO...	0 row(s) affected	0.812 sec
64	15:33:41	CREATE TABLE lab(labID INT PRIMARY KEY, labName VARCHAR(30), zipcode INT ...	0 row(s) affected	0.718 sec
65	15:33:41	CREATE TABLE test(testID INT PRIMARY KEY, testType ENUM('Antibody', 'RT-PCR'), t...	0 row(s) affected	0.844 sec
66	15:33:42	CREATE TABLE symptom(symptomID INT PRIMARY KEY, symptomName ENUM('...	0 row(s) affected	0.469 sec
67	15:33:43	CREATE TABLE surveySymptom (symptomID INT, surveyID INT, CONSTRAINT s...	0 row(s) affected	1.156 sec
68	15:33:44	CREATE TABLE notification(notificationID INT PRIMARY KEY, infectionRisk ENUM('H...	0 row(s) affected	0.812 sec

Fig 7.1: Schema creation in MySQL

CREATE Statements

```
CREATE DATABASE contactTracer;
```

```
USE contactTracer;
```

```
CREATE TABLE appUser(  
  userID INT PRIMARY KEY,  
  userPassword VARCHAR(30) NOT NULL,  
  firstName CHAR(30) NOT NULL,  
  lastName CHAR(30),  
  phone VARCHAR(13) NOT NULL UNIQUE  
);
```

```
CREATE TABLE address(  
  zipcode INT PRIMARY KEY,  
  city CHAR(20) NOT NULL,  
  state CHAR(20) NOT NULL  
);
```

```
CREATE TABLE healthMonitorAdmin(  
  adminID INT PRIMARY KEY,  
  regLat DECIMAL(10,8) signed NOT NULL,  
  regLong DECIMAL(11,8) signed NOT NULL,  
  qualification VARCHAR(30),  
  CONSTRAINT hm_admin FOREIGN KEY(adminID) REFERENCES appUser(userID)  
);
```

```
CREATE TABLE naiveUser(  
  naiveUserID INT PRIMARY KEY,  
  gender ENUM('Male', 'Female', 'Other'),  
  dob DATE NOT NULL,  
  immunoCompromised BOOLEAN NOT NULL,  
  lastSync DATETIME,  
  livingSituation ENUM('Group', 'isolated'),  
  infectionStatus ENUM('infected', 'suspected', 'not-infected') NOT NULL,  
  zipcode INT NOT NULL,  
  CONSTRAINT naivU FOREIGN KEY(naiveUserID) REFERENCES appUser(userID),  
  CONSTRAINT zip FOREIGN KEY(zipcode) REFERENCES address(zipcode)  
);
```

```
CREATE TABLE GPSlocation(  
  gpsLoc INT PRIMARY KEY,  
  gpsLat DECIMAL(10,8) signed NOT NULL,  
  gpsLong DECIMAL(11,8) signed NOT NULL,  
  date DATE NOT NULL  
);
```

```
CREATE TABLE tracker(  
  trackerID INT PRIMARY KEY,  
  gpsLoc INT NOT NULL,  
  naiveUserID INT NOT NULL,  
  CONSTRAINT gps_track FOREIGN KEY(gpsLoc) REFERENCES GPSlocation(gpsLoc),  
  CONSTRAINT track_naive FOREIGN KEY(naiveUserID) REFERENCES naiveUser(naiveUserID)  
);
```

```
CREATE TABLE travelHistory(  
  country VARCHAR(20),  
  naiveUserID INT,  
  CONSTRAINT cn_naive PRIMARY KEY (country, naiveUserID),  
  CONSTRAINT travel_naive FOREIGN KEY(naiveUserID) REFERENCES naiveUser(naiveUserID)  
);
```

```
CREATE TABLE survey(  
  surveyID INT PRIMARY KEY,  
  temperature INT,  
  maskWorn BOOLEAN NOT NULL,  
  dateOfSurvey DATE NOT NULL,  
  naiveUserID INT NOT NULL,  
  CONSTRAINT survey_user FOREIGN KEY(naiveUserID) REFERENCES naiveUser(naiveUserID)  
);
```

```
CREATE TABLE lab(  
  labID INT PRIMARY KEY,  
  labName VARCHAR(30),  
  zipcode INT NOT NULL,  
  CONSTRAINT zip_lab FOREIGN KEY(zipcode) REFERENCES address(zipcode)  
);
```

```
CREATE TABLE test(  
  testID INT PRIMARY KEY,  
  testType ENUM('Antibody','RT-PCR'),  
  testDate DATE NOT NULL,  
  result BOOLEAN NOT NULL,  
  naiveUserID INT NOT NULL,  
  labID INT NOT NULL,  
  CONSTRAINT test_lab FOREIGN KEY(labID) REFERENCES lab(labID),  
  CONSTRAINT test_user FOREIGN KEY(naiveUserID) REFERENCES naiveUser(naiveUserID)  
);
```

```
CREATE TABLE symptom(  
  symptomID INT PRIMARY KEY,  
  symptomName ENUM('Fever', 'Dry Cough', 'Chest Pain', 'Shortness of Breath', 'tiredness')  
);
```

```
CREATE TABLE surveySymptom (  
  symptomID INT,  
  surveyID INT,  
  CONSTRAINT sy_sr PRIMARY KEY(symptomID,surveyID),  
  CONSTRAINT sym_sur FOREIGN KEY(symptomID) REFERENCES symptom(symptomID),  
  CONSTRAINT sur_sym FOREIGN KEY(surveyID) REFERENCES survey(surveyID)  
);
```

```
CREATE TABLE notification(  
  notificationID INT PRIMARY KEY,  
  infectionRisk ENUM('High','loW'),  
  userID INT NOT NULL,  
  CONSTRAINT notif_user FOREIGN KEY(NotificationID) REFERENCES appUser(userID)  
);
```

8.Constraint Checks

Check to ensure that none of the constraints are violated.

```

1  /* 1) constraint check for unique phone number */
2  • INSERT INTO appUser(userID,userPassword,firstName,lastName,phone)
3  VALUES
4  (57,"AOY71IUZ3FQ","Charde","Morse","16460329 5595");
5
6  /* 2) constraint check for unique primary key */
7  • INSERT INTO appUser(userID,userPassword,firstName,lastName,phone)
8  VALUES
9  (2,"THN65TJS3UA","Kendall","Freeman","16110422 8362");
10
11 /* 3) constraint check for ENUM – should fail as "Medium" is not part of the list of permitted values */
12 • INSERT INTO notification (notificationID,infectionRisk,userID) VALUES (875,"Medium",43);
13
14 /* 4) constraint check for referencing non-existent foreign key */
15 • INSERT INTO lab (labID,labName,zipCode)
16 VALUES (51,"In","29778847");
17
18 /* 5) NOT NULL constraint check */
19 • INSERT INTO address (zipcode,city,state)
20 VALUES ("16503",NULL,"MD");
21
22 /* 6) data type violation */
23 • INSERT INTO address (zipcode,city,state)
24 VALUES ("Hello","Gaithersburgg","MDD");

```

100%	40:24
Action Output	
T	Action
1	2) INSERT INTO appUser(userID,userPassword,firstName,lastName,phone) VALUES (57,"AOY71IUZ3FQ","Charde","Morse","16460329 5595"); Error Code: 1062. Duplicate entry '16460329 5595' for key 'appuser.phone'
2	INSERT INTO appUser(userID,userPassword,firstName,lastName,phone) VALUES (2,"THN65TJS3UA","Kendall","Freeman","16110422 8362"); Error Code: 1062. Duplicate entry '2' for key 'appuser.PRIMARY'
3	INSERT INTO notification (notificationID,infectionRisk,userID) VALUES (875,"Medium",43); Error Code: 1265. Data truncated for column 'infectionRisk' at row 1
4	INSERT INTO lab (labID,labName,zipCode) VALUES (51,"In","29778847"); Error Code: 1452. Cannot add or update a child row: a foreign key constraint fails ('contacttracer'.lab', CONSTRAINT 'zip_lab' FOREIGN KEY ('zipcode') REFERENCES 'address' ('zipcode'))
5	INSERT INTO address (zipcode,city,state) VALUES ("16503",NULL,"MD"); Error Code: 1048. Column 'city' cannot be null
6	2) INSERT INTO address (zipcode,city,state) VALUES ("Hello","Gaithersburgg","MDD"); Error Code: 1366. Incorrect integer value: 'Hello' for column 'zipcode' at row 1

Fig 8.1: Proof of constraint checks

9.Inserting Data into database

Relevant dummy data inserted into the database using www.generatedata.com

Query 1

<

Fig 9.1: Sample insert statements in MySQL

Query 1 naiveUser

Limit to 1000 rows

1 • **SELECT * FROM** contactTracer.naiveUser;

100% 39:1

Result Grid Filter Rows: Search Edit: Export/Import:

naiveUserID	gender	dob	immunoCompromis...	lastSync	livingSituati...	infectionStat...	zipcode
3	Male	1971-06-13	1	2020-09-28 03:06:18	Group	not-infected	46833
4	Male	1993-05-19	1	2021-01-26 07:14:09	isolated	not-infected	72539
5	Female	2005-06-06	0	2020-04-17 06:15:19	isolated	not-infected	99829
6	Male	1989-07-25	1	2019-08-05 20:08:26	Group	not-infected	75508
7	Male	2065-01-30	1	2021-05-14 22:38:24	isolated	suspected	46833
8	Male	2051-09-26	0	2020-03-08 04:02:15	Group	not-infected	75508
9	Female	2012-09-18	1	2021-03-04 11:45:41	isolated	not-infected	75508
10	Male	1987-02-14	1	2019-11-16 11:49:35	Group	not-infected	72580
11	Female	1995-04-18	1	2020-09-22 16:54:18	Group	infected	46833
12	Male	2061-12-31	0	2020-07-18 02:15:20	Group	not-infected	15278
13	Male	1974-04-30	0	2019-10-10 15:28:43	isolated	not-infected	15278
14	Male	1986-10-06	1	2020-05-15 05:36:23	isolated	suspected	15278
15	Male	1998-04-09	1	2020-12-05 03:56:32	isolated	infected	46833
16	Male	2000-12-26	0	2020-06-09 01:10:37	Group	infected	15278
17	Female	2006-12-30	0	2020-04-27 01:41:20	Group	not-infected	15278
18	Female	2059-04-19	0	2020-09-23 19:59:03	Group	not-infected	15278
19	Female	1988-10-07	1	2020-07-04 09:33:26	isolated	not-infected	15278
20	Male	1980-08-24	0	2021-01-05 17:10:06	Group	infected	46833
21	Male	1972-07-31	0	2020-01-20 18:43:14	isolated	not-infected	72580
22	Female	2051-06-24	1	2019-11-22 13:55:43	isolated	not-infected	72580
23	Female	2003-09-28	0	2020-07-03 03:27:40	isolated	not-infected	86995
24	Male	2052-02-22	0	2020-07-01 18:27:18	isolated	suspected	86995
25	Male	1972-11-19	0	2019-11-22 15:44:44	isolated	suspected	46833
26	Male	2068-12-07	1	2020-12-30 22:50:06	isolated	suspected	86995
27	Female	1983-10-01	0	2021-01-22 18:46:40	Group	suspected	86995
28	Male	2013-08-13	0	2020-08-11 16:42:35	Group	not-infected	31365
29	Female	1970-09-20	0	2020-03-16 17:11:34	isolated	infected	31365
30	Female	1994-02-02	0	2020-09-19 04:01:44	Group	not-infected	31365
31	Male	2066-10-07	0	2021-02-21 23:15:13	isolated	not-infected	31365
32	Male	2068-06-30	1	2020-02-19 16:54:31	Group	infected	42042
33	Female	2053-02-13	1	2021-01-20 09:10:00	isolated	not-infected	28270
34	Male	2057-06-11	1	2021-05-21 02:33:10	Group	not-infected	28270
35	Female	2053-10-13	1	2019-12-31 07:43:27	Group	suspected	28270
36	Female	2060-05-04	1	2019-10-23 07:14:01	isolated	not-infected	21421

naiveUser 2

Action Output

	Time	Action
1	19:57:53	SELECT * FROM contactTracer.naiveUser LIMIT 0, 1000

Fig 9.2: Result of Select * from table naiveUser

10.Load Data into R

```
## [1] "address"          "appuser"          "gpslocation"
## [4] "healthmonitoradmin" "lab"              "naiveuser"
## [7] "notification"      "survey"           "surveysymptom"
## [10] "symptom"           "test"             "tracker"
## [13] "travelhistory"
```

11. Execute Queries in R to show ContactTrace database

1.(Join of 3 tables) Names of users from cities Grand Island, Harrisburg and Fayetteville who have a travel history to any country

```
query1<- "SELECT appUser.firstName , appUser.lastName, address.city, travelHistory.country
'Country travelled to'
FROM appUser,NaiveUser,travelHistory, address
WHERE appUser.userID = naiveUser.naiveUserID AND
naiveUser.naiveUserID= travelHistory.naiveUserID AND
address.zipcode=naiveUser.zipcode AND
address.city IN ('Grand Island','Fayetteville','Harrisburg');"
```

##	firstName	lastName	city	Country travelled to
## 1	Allistair	Conway	Grand Island	Japan
## 2	Maile	Lott	Grand Island	Japan
## 3	Ira	Leon	Grand Island	Uganda
## 4	Alden	Carson	Grand Island	Sweden
## 5	Aiko	Barrett	Grand Island	Kuwait
## 6	Allen	Duke	Grand Island	Sweden
## 7	Upton	Bradley	Grand Island	Botswana
## 8	Rashad	Jackson	Fayetteville	Uganda
## 9	Chastity	Bruce	Fayetteville	Andorra
## 10	Kessie	Pena	Fayetteville	Sweden
## 11	Christen	Bowen	Harrisburg	Guinea-Bissau
## 12	Wyatt	Lancaster	Harrisburg	Sweden
## 13	Judah	Sims	Harrisburg	Nauru
## 14	Keith	Hess	Harrisburg	Australia

2.(Subquery) Find non-infected users and their living situation who have come in contact with a infected person (**CONTACT TRACING**) based on GPS location and date

```
query2<-"SELECT distinct naiveUser.naiveUserID, naiveUser.livingSituation
FROM naiveUser, tracker, GPSlocation
WHERE naiveUser.naiveUserID = tracker.naiveUserID
AND tracker.gpsLoc = GPSlocation.gpsLoc
AND naiveUser.infectionStatus != 'infected'
AND tracker.gpsLoc IN ( SELECT tracker.gpsLoc FROM naiveUser, tracker
WHERE naiveUser.naiveUserID = tracker.naiveUserID
AND naiveUser.infectionStatus = 'infected');"
```

```
##      naiveUserID livingSituation
## 1           3           Group
## 2           4           isolated
## 3           5           isolated
## 4           7           isolated
## 5          10           Group
## 6          14           isolated
## 7          19           isolated
## 8          23           isolated
## 9          24           isolated
## 10         36           isolated
## 11         40           Group
## 12         46           isolated
## 13         52           Group
```

3.(Use having and Group By) List of all labs which have conducted more than 2 tests

```
query3<-"SELECT Count(test.testID) AS 'Test Count', lab.labName 'Lab Name'
FROM lab,test
WHERE test.labID = lab.labID
GROUP BY lab.labID
HAVING COUNT(test.testID)>1 ;"
```

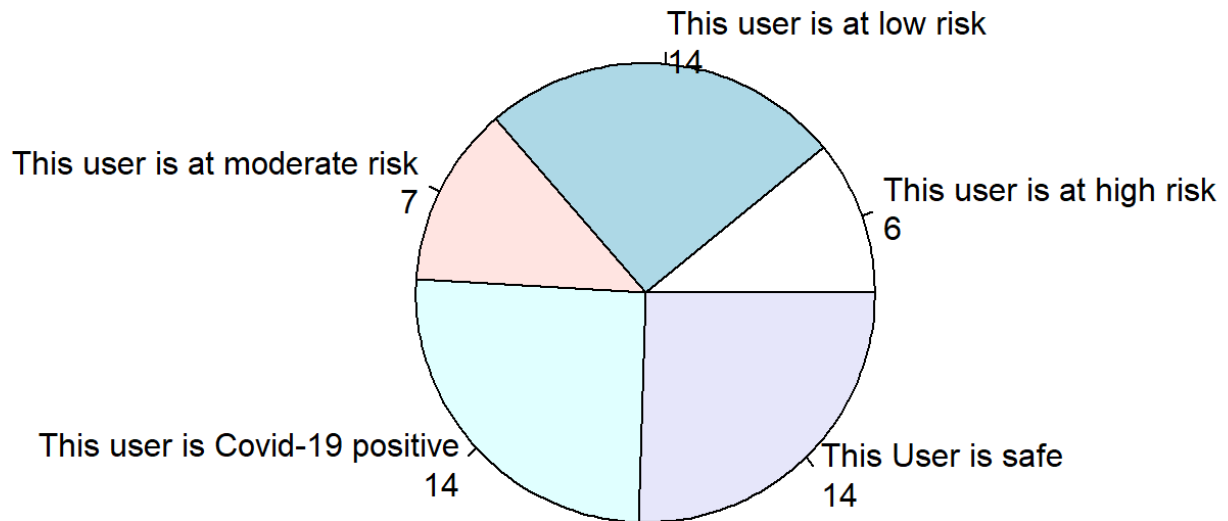
```
##      Test Count  Lab Name
## 1           2      mus
## 2           2      lacus
## 3           2      nisl
## 4           2      mollis
## 5           2       at
## 6           2      cursus
## 7           2      etpoy
## 8           2 malesuada
```

4.(Complex query using SELECT CASE/WHEN)To determine risk status of a person

```
query4<-"SELECT naiveUserID, infectionStatus, immunoCompromised,
CASE WHEN infectionStatus = 'suspected' AND immunoCompromised = TRUE THEN 'This user is at
high risk'
WHEN infectionStatus = 'suspected' AND immunoCompromised = FALSE THEN 'This user is at mode
rate risk'
WHEN infectionStatus = 'infected' THEN 'This user is Covid-19 positive'
WHEN infectionStatus = 'not-infected' AND immunoCompromised = TRUE THEN 'This user is at lo
w risk'
ELSE 'This User is safe'
END AS Alert
FROM naiveUser;"
```

##	naiveUserID	infectionStatus	immunoCompromised	Alert
## 1	1	infected	1	This user is Covid-19 positive
## 2	2	suspected	1	This user is at high risk
## 3	3	not-infected	1	This user is at low risk
## 4	4	not-infected	1	This user is at low risk
## 5	5	not-infected	0	This User is safe
## 6	6	not-infected	1	This user is at low risk
## 7	7	suspected	1	This user is at high risk
## 8	8	not-infected	0	This User is safe
## 9	9	not-infected	1	This user is at low risk
## 10	10	not-infected	1	This user is at low risk
## 11	11	infected	1	This user is Covid-19 positive
## 12	12	not-infected	0	This User is safe
## 13	13	not-infected	0	This User is safe
## 14	14	suspected	1	This user is at high risk
## 15	15	infected	1	This user is Covid-19 positive
## 16	16	infected	0	This user is Covid-19 positive
## 17	17	not-infected	0	This User is safe
## 18	18	not-infected	0	This User is safe
## 19	19	not-infected	1	This user is at low risk
## 20	20	infected	0	This user is Covid-19 positive
## 21	21	not-infected	0	This User is safe
## 22	22	not-infected	1	This user is at low risk
## 23	23	not-infected	0	This User is safe
## 24	24	suspected	0	This user is at moderate risk
## 25	25	suspected	0	This user is at moderate risk
## 26	26	suspected	1	This user is at high risk
## 27	27	suspected	0	This user is at moderate risk
## 28	28	not-infected	0	This User is safe
## 29	29	infected	0	This user is Covid-19 positive
## 30	30	not-infected	0	This User is safe
## 31	31	not-infected	0	This User is safe
## 32	32	infected	1	This user is Covid-19 positive
## 33	33	not-infected	1	This user is at low risk
## 34	34	not-infected	1	This user is at low risk
## 35	35	suspected	1	This user is at high risk
## 36	36	not-infected	1	This user is at low risk
## 37	37	suspected	0	This user is at moderate risk
## 38	38	not-infected	0	This User is safe
## 39	39	not-infected	0	This User is safe
## 40	40	not-infected	1	This user is at low risk
## 41	41	not-infected	0	This User is safe
## 42	42	infected	0	This user is Covid-19 positive
## 43	43	infected	1	This user is Covid-19 positive
## 44	44	not-infected	1	This user is at low risk
## 45	45	infected	0	This user is Covid-19 positive
## 46	46	suspected	1	This user is at high risk
## 47	47	infected	1	This user is Covid-19 positive
## 48	48	infected	1	This user is Covid-19 positive
## 49	49	suspected	0	This user is at moderate risk
## 50	50	infected	1	This user is Covid-19 positive
## 51	51	infected	0	This user is Covid-19 positive
## 52	52	suspected	0	This user is at moderate risk
## 53	53	not-infected	1	This user is at low risk
## 54	54	suspected	0	This user is at moderate risk
## 55	55	not-infected	1	This user is at low risk

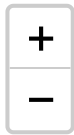
Pie-chart depicting user's risk of being infected by Covid-19



5.(Query of choice) GPS coordinates of infected users for Plotting- dummy data

```
query5<-"SELECT gpslocation.gpsLat AS Latitude, gpslocation.gpsLong AS Longitude
FROM naiveUser, tracker, gpslocation
WHERE naiveUser.naiveUserID = tracker.naiveUserID AND
tracker.gpsLoc = gpslocation.gpsLoc
AND naiveUser.infectionStatus = 'infected';"
```

```
##      Latitude  Longitude
## 1    47.40403   131.83584
## 2    22.38534    35.53054
## 3    52.10685   142.92803
## 4    86.66287    22.99641
## 5    47.40403   131.83584
## 6     1.88978   -21.19278
## 7   -86.95076  -131.54891
## 8   -56.85797    94.08396
## 9    36.54954  -157.57636
## 10   39.21384   -66.08290
## 11  -26.39023   139.71786
## 12  -85.96642  -127.08869
## 13  -87.20021    36.18430
## 14   64.46163   104.56520
## 15  -27.58789    68.46440
## 16  -31.69499    24.58768
## 17  -27.58789    68.46440
```



Leaflet (<http://leafletjs.com>) | © OpenStreetMap (<http://openstreetmap.org>) contributors, CC-BY-SA
(<http://creativecommons.org/licenses/by-sa/2.0/>)

6.(optional) Select of number of infected cases per city

```
query7<-"SELECT count(naiveUser.naiveUserID) AS 'infected case count', address.city
FROM naiveUser, address
WHERE naiveUser.zipcode= address.zipcode AND
naiveUser.infectionStatus = 'infected'
GROUP BY(address.city);
"
```

##	infected case count	city
## 1	7	Flint
## 2	1	Grand Island
## 3	2	Helena
## 4	1	Kapolei
## 5	1	New Orleans
## 6	1	San Antonio
## 7	1	South Portland