SMBUD 2021 - Project work

NoSQL DB for supporting a contact tracing application for COVID-19.

Authors:

Pavithra Saravana Moorthy - 10748677
Vinaykrishna Munnaluri - 10780972
Stancioi Ioana-Ruxandra - 10855466
Roger Aldair Veliz Sedano - 10614456
Wilson Emannuelle Quispe Limay - 10182182

Contents

1 Introduction

The COVID-19 pandemic outbreak has offered significant challenges for organizations and governments.
Our resulting model could allow several stakeholders (collaboration of researchers, software developers, data scientists, medical professionals and general users) to build applications in order to support the data analyses that can help the COVID-19 disease mitigation and keep their country safe.

1.1 Problem formulation & Solution

We start designing our graph database solution by gathering facts and requirements to solve the pandemic problem of contact tracing. For this reason we defined a series of questions that guided the problem's conceptualization:

Q1. How to track people?
Q3. How can we describe the contacts between people?
Q4. How to monitor the people who come into contact with each other?

To answer the first question, we collect open-source data using different technologies such as API or web scraping. However, for more accurate testing purposes we decided to generate our own data. For each person, we considered the demographic data, the clinical data, the geolocation data and the timestamp data. In particular the last two are useful when a person tests positive for COVID-19 and we want to notify people that were in contact with him/her during the last days.

As for Q2, the question can be addressed by defining 2 types of contact: permanent contact which are people or families living in the same house and occasional contacts when people go to public places such as hospitals, workplaces, restaurants, cinemas, theaters or when people visit other households. For this testing phase, we don't consider card transportation to track people in the same bus or underground and travel tickets for trains or flights.

The last question can be answered by taking into consideration the different types of connections between people. For occasional contact we evaluate geolocation and timestamp data generated by smartphones every 30 minutes. Explicit data (telephone, contact name) provided by people when entering a public location could be available depending on the privacy regulation for each country and if there's any digital storage for a reasonable time period. We don't consider credit card information, it could be useful in case we need to track people in the same restaurant, theater, cinema, shopping centers when they pay with their debit or credit card. For permanent contact we consider by default that they are always in touch, therefore when and/or where information is not really necessary.

2 Data Model

2.1 Design

To explain and illustrate our model, we employ the Entity Relationship (ER) diagram, which allows building abstractions of real objects and occurrences, such as classification, generalization and aggregation.

2.2 ER Diagram

In the following links we provide the ER diagram details and list of attributes and their corresponding entity.

- Link - ER diagram detail
- Link - Entity list and its description

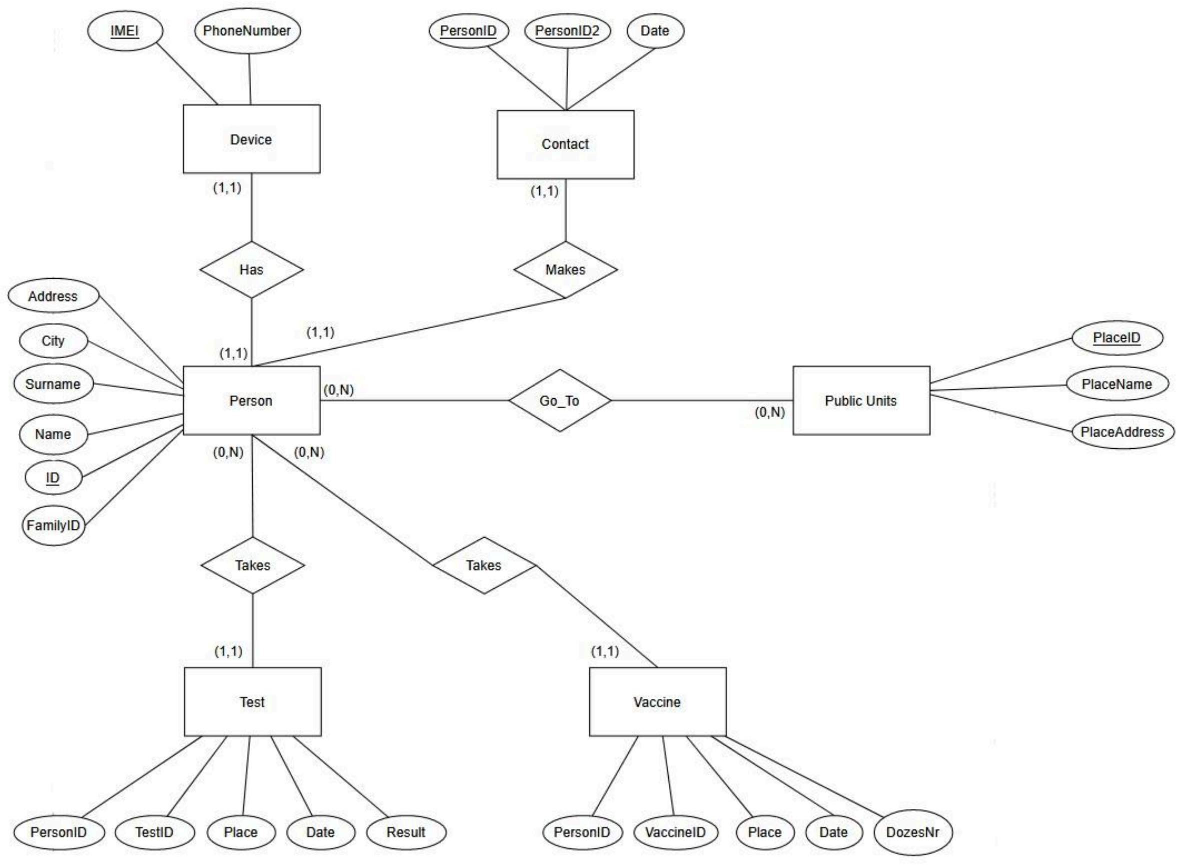


Fig. 1. Entity-Relationship diagram of contact tracing solution.

3 Implementation

Data generated using fake random data generator websites such as https://www.onlinedatagenerator.com/ or https://www.mockaroo.com/
Data dimension: 1000 People
People data file Link: data.csv
Graph Database file Link: neo4j_db

3.1 Neo4j Implementation

Neo4j is an open-source native graph database that implements the property graph model. Cypher query language is the declarative language used for the Neo4j graph database.

3.1.1 Nodes

1) Person, example:

```
{
  "identity": 4,
  "labels": [
    "Person"
  ],
  "properties": {
"name": "Summer",
"address": "Carltoun   Tunnel,
6927",
"phone": "8-863-516-7182",
"city": "Fort Lauderdale",
"surname": "Summers"
  }
}
```

2) Restaurant, example:

```
{
  "identity": 1578,
  "labels": [
    "Restaurants"
  ],
  "properties": {
"name": "Bello Italiano",
"address": "Wakley Crossroad,
4340",
"postalcode": "4340"
  }
}
```

3) Covidtest, example:

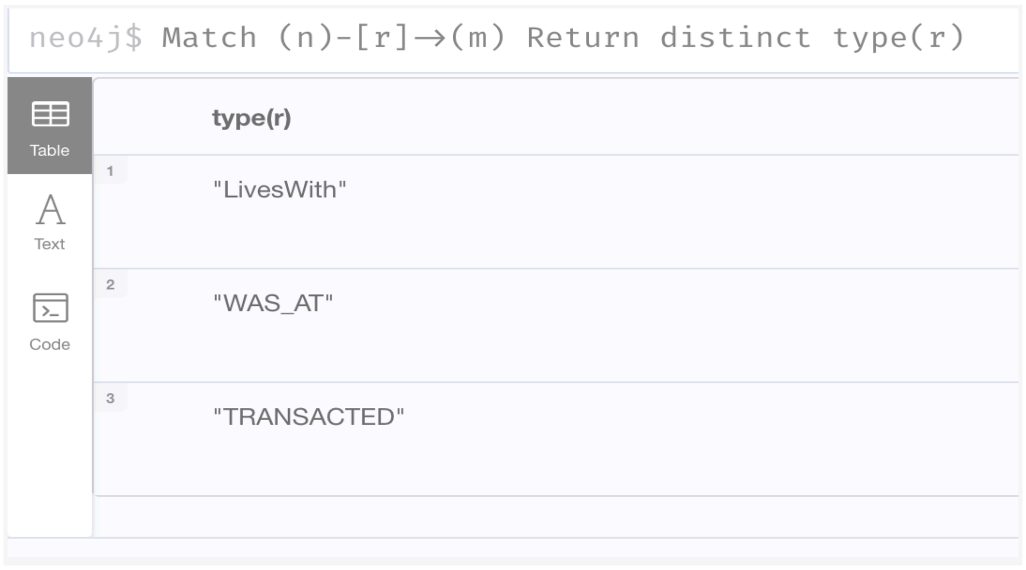
```
{
  "identity": 1502,
  "labels": [
    "COVIDTEST"
  ],
  "properties": {
"result": "negative",
"name": "Owen",
"phone": "3-566-554-0212",
"testedon":
"2021-11-11T10:40:32.142000000+01
:00",
"surname": "Mooney"
  }
}
```

4) Tower(Cell phone Tower), example:

```
{
"identity": 1010,
"labels": [
"Tower"
],
"properties": {
"name": "Tower1"
 }
}
```

Tab. 1. Types of nodes

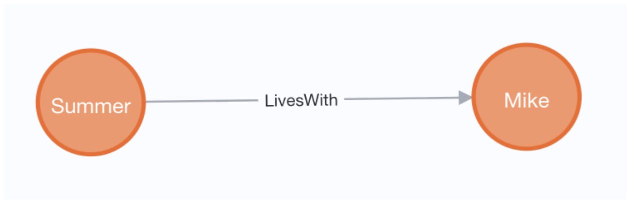
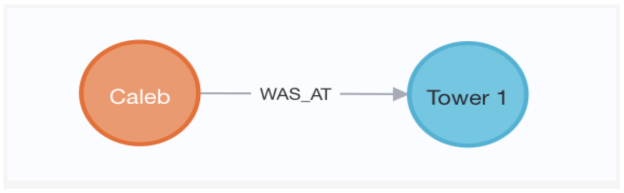
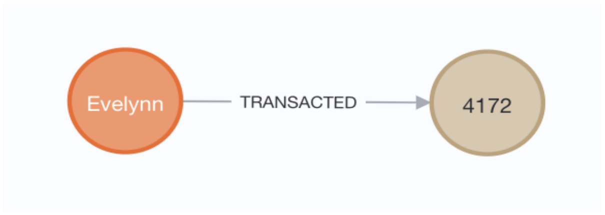
3.1.2 Relationships



```
neo4j$ Match (n)-[r]→(m) Return distinct type(r)
```

	type(r)
1	"LivesWith"
2	"WAS_AT"
3	"TRANSACTED"

Fig. 2. Types of relationships

1) Summer LivesWith Mike 
2) Caleb WAS_AT Tower (Cell phone Tower) 
3) Evelynn TRANSACTED Restaurant 

Tab. 2. Examples of relationships

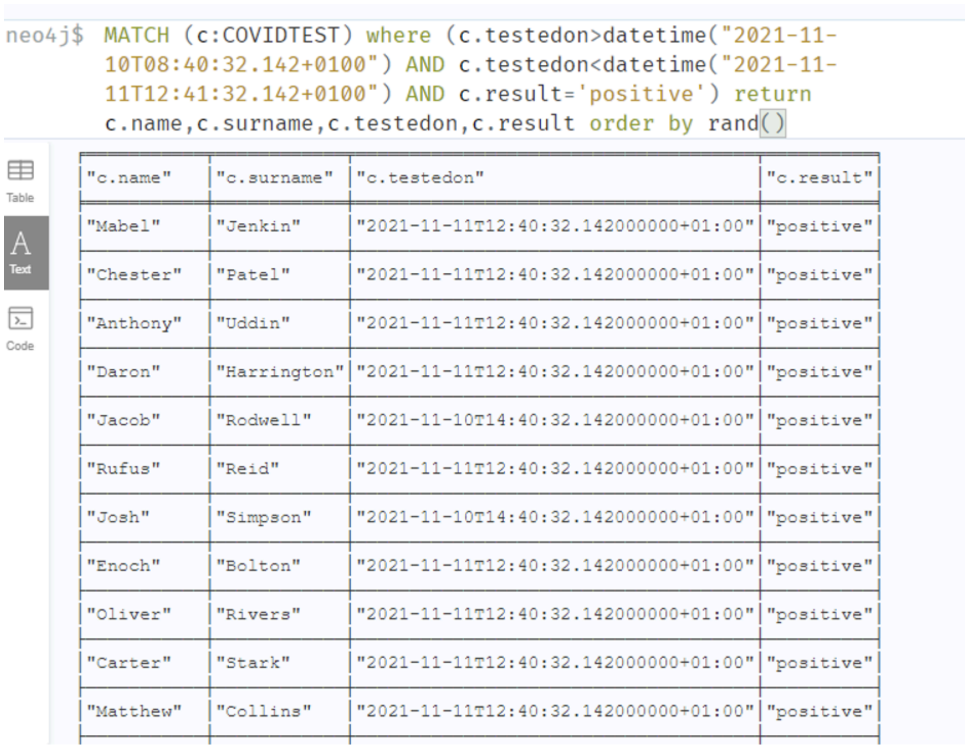
3.2 Queries

List of queries using Cypher query language:

1) Find positive tested people between a specific time period

Query used to retrieve name, surname of people tested positive between a specific time period.

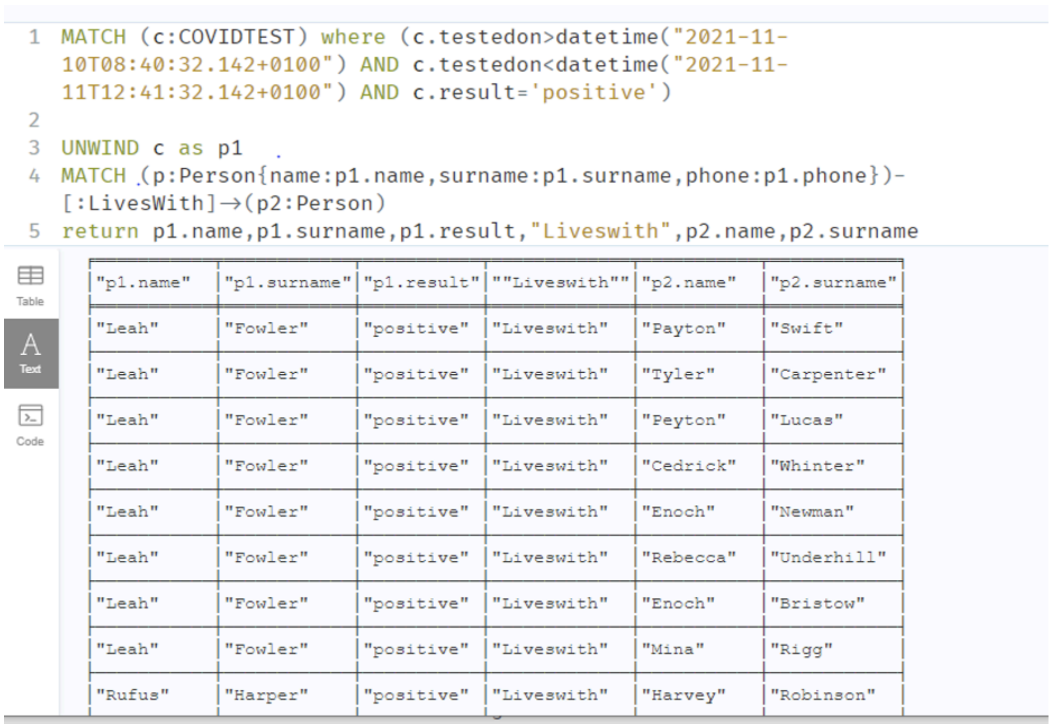
```
MATCH (c:COVIDTEST) where (c.testedon>datetime("2021-11-10T08:40:32.142+0100") AND
c.testedon<datetime("2021-11-11T12:41:32.142+0100") AND c.result='positive') return
c.name,c.surname,c.testedon,c.result order by rand()
```



```
neo4j$ MATCH (c:COVIDTEST) where (c.testedon>datetime("2021-11-
       10T08:40:32.142+0100") AND c.testedon<datetime("2021-11-
       11T12:41:32.142+0100") AND c.result='positive') return
       c.name,c.surname,c.testedon,c.result order by rand()
```

```
│"c.name"   │"c.surname" │"c.testedon"                         │"c.result"│
│"Mabel"    │"Jenkin"    │"2021-11-11T12:40:32.142000000+01:00"│"positive"│
│"Chester"  │"Patel"     │"2021-11-11T12:40:32.142000000+01:00"│"positive"│
│"Anthony"  │"Uddin"     │"2021-11-11T12:40:32.142000000+01:00"│"positive"│
│"Daron"    │"Harrington"│"2021-11-11T12:40:32.142000000+01:00"│"positive"│
│"Jacob"    │"Rodwell"   │"2021-11-10T14:40:32.142000000+01:00"│"positive"│
│"Rufus"    │"Reid"      │"2021-11-11T12:40:32.142000000+01:00"│"positive"│
│"Josh"     │"Simpson"   │"2021-11-10T14:40:32.142000000+01:00"│"positive"│
│"Enoch"    │"Bolton"    │"2021-11-11T12:40:32.142000000+01:00"│"positive"│
│"Oliver"   │"Rivers"    │"2021-11-11T12:40:32.142000000+01:00"│"positive"│
│"Carter"   │"Stark"     │"2021-11-11T12:40:32.142000000+01:00"│"positive"│
│"Matthew"  │"Collins"   │"2021-11-11T12:40:32.142000000+01:00"│"positive"│
```

2) *Infected people with their cohabitee*

Query used to retrieve name, surname, phone number of people tested positive between a specific time period and the name, surname of their cohabitee / family members (LivesWith).
MATCH (c:COVIDTEST) where (c.testedon>datetime("2021-11-10T08:40:32.142+0100") AND c.testedon<datetime("2021-11-11T12:41:32.142+0100") AND c.result='positive') UNWIND c as p1 MATCH (p:Person{name:p1.name,surname:p1.surname,phone:p1.phone})-[:LivesWith]->(p2:Person) return p1.name,p1.surname,p1.result,"Liveswith",p2.name,p2.surname


```
MATCH (c:COVIDTEST) where (c.testedon>datetime("2021-11-
10T08:40:32.142+0100") AND c.testedon<datetime("2021-11-
11T12:41:32.142+0100") AND c.result='positive')

UNWIND c as p1
MATCH (p:Person{name:p1.name,surname:p1.surname,phone:p1.phone})-
[:LivesWith]→(p2:Person)
return p1.name,p1.surname,p1.result,"Liveswith",p2.name,p2.surname
```

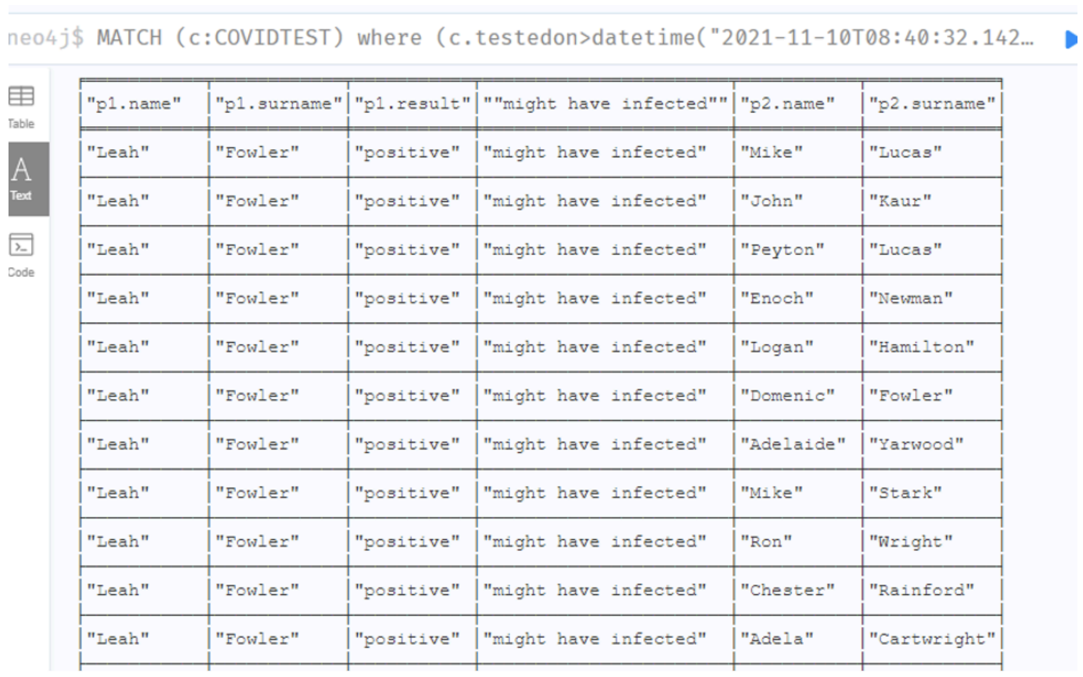
```
╒═════════╤════════════╤═══════════╤═════════════╤═══════════╤═════════════╕
│"p1.name"│"p1.surname"│"p1.result"│""Liveswith""│"p2.name"  │"p2.surname" │
╞═════════╪════════════╪═══════════╪═════════════╪═══════════╪═════════════╡
│"Leah"   │"Fowler"    │"positive" │"Liveswith"  │"Payton"   │"Swift"      │
├─────────┼────────────┼───────────┼─────────────┼───────────┼─────────────┤
│"Leah"   │"Fowler"    │"positive" │"Liveswith"  │"Tyler"    │"Carpenter"  │
├─────────┼────────────┼───────────┼─────────────┼───────────┼─────────────┤
│"Leah"   │"Fowler"    │"positive" │"Liveswith"  │"Peyton"   │"Lucas"      │
├─────────┼────────────┼───────────┼─────────────┼───────────┼─────────────┤
│"Leah"   │"Fowler"    │"positive" │"Liveswith"  │"Cedrick"  │"Whinter"    │
├─────────┼────────────┼───────────┼─────────────┼───────────┼─────────────┤
│"Leah"   │"Fowler"    │"positive" │"Liveswith"  │"Enoch"    │"Newman"     │
├─────────┼────────────┼───────────┼─────────────┼───────────┼─────────────┤
│"Leah"   │"Fowler"    │"positive" │"Liveswith"  │"Rebecca"  │"Underhill"  │
├─────────┼────────────┼───────────┼─────────────┼───────────┼─────────────┤
│"Leah"   │"Fowler"    │"positive" │"Liveswith"  │"Enoch"    │"Bristow"    │
├─────────┼────────────┼───────────┼─────────────┼───────────┼─────────────┤
│"Leah"   │"Fowler"    │"positive" │"Liveswith"  │"Mina"     │"Rigg"       │
├─────────┼────────────┼───────────┼─────────────┼───────────┼─────────────┤
│"Rufus"  │"Harper"    │"positive" │"Liveswith"  │"Harvey"   │"Robinson"   │
```

3) *Two steps relation, “might have infected”*

Query used to retrieve name, surname, phone number of people tested positive between a specific time period and the name, surname of people that are connected or have been in contact with their cohabitee / family members (LivesWith*2).

```
MATCH (c:COVIDTEST) where (c.testedon>datetime("2021-11-10T08:40:32.142+0100") AND
c.testedon<datetime("2021-11-11T12:41:32.142+0100") AND c.result='positive')

UNWIND c as p1
MATCH
(p:Person{name:p1.name,surname:p1.surname,phone:p1.phone})-[:LivesWith*2]->(p2:Person)
return p1.name,p1.surname,p1.result,"might have infected",p2.name,p2.surname
```



```
neo4j$ MATCH (c:COVIDTEST) where (c.testedon>datetime("2021-11-10T08:40:32.142…

|"p1.name"|"p1.surname"|"p1.result"|""might have infected""|"p2.name" |"p2.surname"|
|"Leah"   |"Fowler"    |"positive" |"might have infected"  |"Mike"    |"Lucas"     |
|"Leah"   |"Fowler"    |"positive" |"might have infected"  |"John"    |"Kaur"      |
|"Leah"   |"Fowler"    |"positive" |"might have infected"  |"Peyton"  |"Lucas"     |
|"Leah"   |"Fowler"    |"positive" |"might have infected"  |"Enoch"   |"Newman"    |
|"Leah"   |"Fowler"    |"positive" |"might have infected"  |"Logan"   |"Hamilton"  |
|"Leah"   |"Fowler"    |"positive" |"might have infected"  |"Domenic" |"Fowler"    |
|"Leah"   |"Fowler"    |"positive" |"might have infected"  |"Adelaide"|"Yarwood"   |
|"Leah"   |"Fowler"    |"positive" |"might have infected"  |"Mike"    |"Stark"     |
|"Leah"   |"Fowler"    |"positive" |"might have infected"  |"Ron"     |"Wright"    |
|"Leah"   |"Fowler"    |"positive" |"might have infected"  |"Chester" |"Rainford"  |
|"Leah"   |"Fowler"    |"positive" |"might have infected"  |"Adela"   |"Cartwright"|
```

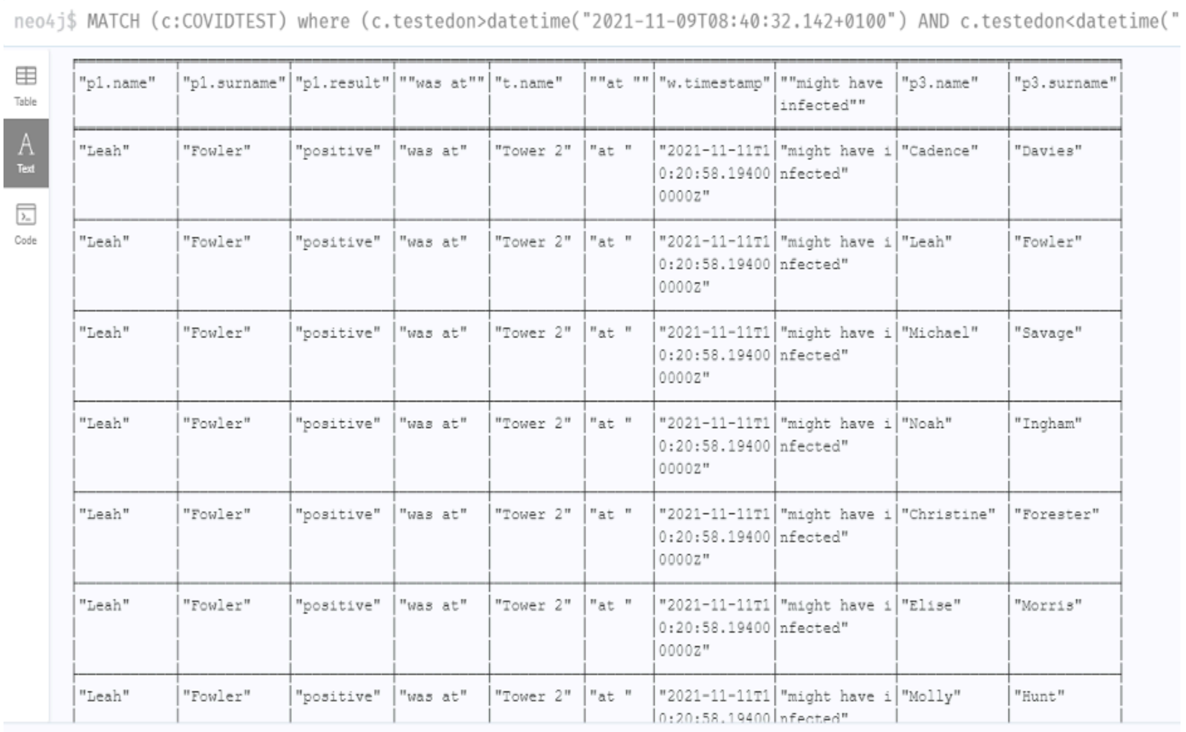
4) *People along the path of infected people, detected using cell towers*

Query used to retrieve name, surname, phone number of people tested positive between a specific time period and the name, surname of people that have been in contact with them based on the same cell tower id (name) and timestamp greater than the timestamp when the people have been tested positive.

```
MATCH (c:COVIDTEST) where (c.testedon>datetime("2021-11-09T08:40:32.142+0100") AND
c.testedon<datetime("2021-11-11T12:41:32.142+0100") AND c.result='positive')

UNWIND c as p1
MATCH
(p:Person{name:p1.name,surname:p1.surname,phone:p1.phone})-[w:WAS_AT]->(t:TOWER)

UNWIND t as t1
 MATCH (p3:Person) -[w2:WAS_AT]->(t2:TOWER{name:t1.name}) where
w2.timestamp>datetime("2021-11-09T08:40:32.142+0100")
return p1.name,p1.surname,p1.result,"was at",t.name,"at ",w.timestamp,"might have
infected",p3.name,p3.surname
```



```
neo4j$ MATCH (c:COVIDTEST) where (c.testedon>datetime("2021-11-09T08:40:32.142+0100") AND c.testedon<datetime("

|"p1.name"|"p1.surname"|"p1.result"|""was at""|"t.name"|""at ""|"w.timestamp"|""might have |"p3.name"  |"p3.surname"|
|         |            |           |          |        |       |             |infected""   |           |            |
|"Leah"   |"Fowler"    |"positive" |"was at"  |"Tower 2"|"at " |"2021-11-11T1|"might have i|"Cadence"  |"Davies"    |
|         |            |           |          |        |       |0:20:58.19400|nfected"     |           |            |
|         |            |           |          |        |       |0000Z"       |             |           |            |
|"Leah"   |"Fowler"    |"positive" |"was at"  |"Tower 2"|"at " |"2021-11-11T1|"might have i|"Leah"     |"Fowler"    |
|         |            |           |          |        |       |0:20:58.19400|nfected"     |           |            |
|         |            |           |          |        |       |0000Z"       |             |           |            |
|"Leah"   |"Fowler"    |"positive" |"was at"  |"Tower 2"|"at " |"2021-11-11T1|"might have i|"Michael"  |"Savage"    |
|         |            |           |          |        |       |0:20:58.19400|nfected"     |           |            |
|         |            |           |          |        |       |0000Z"       |             |           |            |
|"Leah"   |"Fowler"    |"positive" |"was at"  |"Tower 2"|"at " |"2021-11-11T1|"might have i|"Noah"     |"Ingham"    |
|         |            |           |          |        |       |0:20:58.19400|nfected"     |           |            |
|         |            |           |          |        |       |0000Z"       |             |           |            |
|"Leah"   |"Fowler"    |"positive" |"was at"  |"Tower 2"|"at " |"2021-11-11T1|"might have i|"Christine"|"Forester"  |
|         |            |           |          |        |       |0:20:58.19400|nfected"     |           |            |
|         |            |           |          |        |       |0000Z"       |             |           |            |
|"Leah"   |"Fowler"    |"positive" |"was at"  |"Tower 2"|"at " |"2021-11-11T1|"might have i|"Elise"    |"Morris"    |
|         |            |           |          |        |       |0:20:58.19400|nfected"     |           |            |
|         |            |           |          |        |       |0000Z"       |             |           |            |
|"Leah"   |"Fowler"    |"positive" |"was at"  |"Tower 2"|"at " |"2021-11-11T1|"might have i|"Molly"    |"Hunt"      |
|         |            |           |          |        |       |0:20:58.19400|nfected"     |           |            |
```

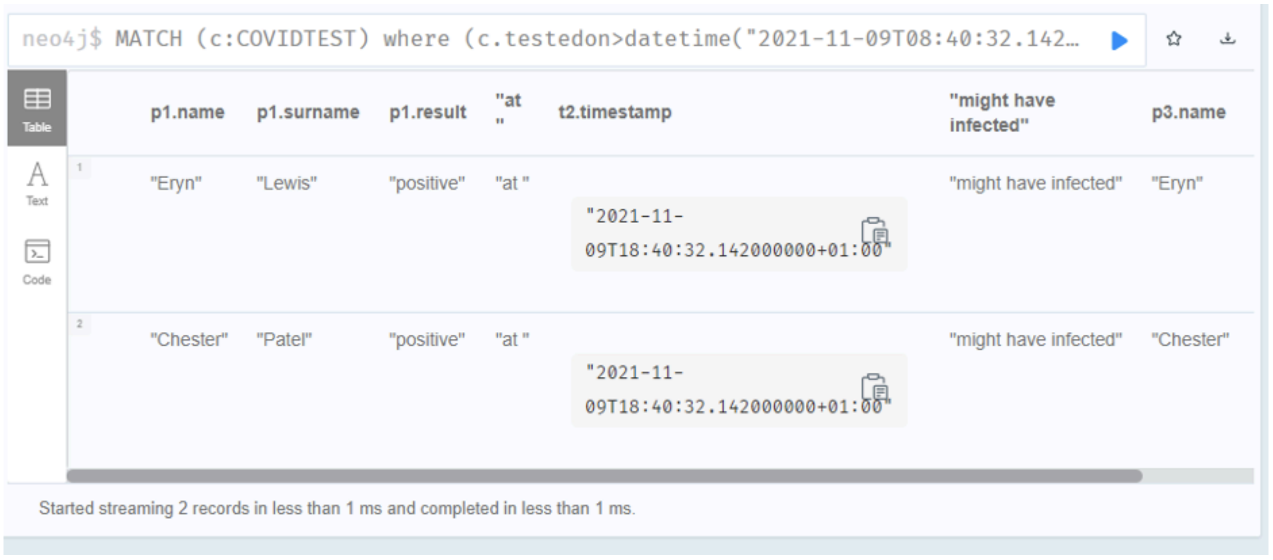
5) *Path along positive tested people, detected using public places (restaurant)*

Query used to retrieve name, surname, phone number of people tested positive between a specific time period and the name, surname of people that have been in contact with them based on the same public place (restaurant) and timestamp greater than the timestamp when the people have been tested positive.

```
MATCH (c:COVIDTEST) where (c.testedon>datetime("2021-11-09T08:40:32.142+0100") AND
c.testedon<datetime("2021-11-11T12:41:32.142+0100") AND c.result='positive')

UNWIND c as p1
MATCH
(p:Person{name:p1.name,surname:p1.surname,phone:p1.phone})-[t:TRANSACTED]->(r:Restaur
ants)
UNWIND r as r1

MATCH (p3:Person) -[t2:TRANSACTED]->(r1) where
t2.timestamp>datetime("2021-11-09T08:40:32.142+0100")
return p1.name,p1.surname,p1.result,"at ",t2.timestamp,"might have
infected",p3.name,p3.surname
```



```
neo4j$ MATCH (c:COVIDTEST) where (c.testedon>datetime("2021-11-09T08:40:32.142…
```

	p1.name	p1.surname	p1.result	"at "	t2.timestamp	"might have infected"	p3.name
1	"Eryn"	"Lewis"	"positive"	"at "	"2021-11-09T18:40:32.142000000+01:00"	"might have infected"	"Eryn"
2	"Chester"	"Patel"	"positive"	"at "	"2021-11-09T18:40:32.142000000+01:00"	"might have infected"	"Chester"

Started streaming 2 records in less than 1 ms and completed in less than 1 ms.

3.3 Commands

1) *Command for loading data from csv file (restaurants.csv)*

```
LOAD CSV FROM 'file:///restaurants..csv' AS line
CREATE(:Restaurants{name:line[0],address:line[1],postalcode:line[2]})
```

2) *Command for generating random “LivesWith” relationships based on probability*

```
MATCH (p12:Person)   WITH  p12 order by rand() LIMIT 100

UNWIND p12 AS p1

MATCH (p22:Person)   WITH  p22 order by rand() LIMIT 100

UNWIND p22 AS p2

MATCH (p2),(p1) WITH p1, p2 where p1<>p2 AND rand()<0.1 and not
exists((p1)-[:LivesWith]->(p2)) MERGE (p1)-[:LivesWith]->(p2)
```

3) *Command for generating data in order to create people tested positive also visited restaurant*

```
MATCH (p12:Person)   WITH  p12 order by rand() LIMIT 5
unwind p12 as p
create
(c:COVIDTEST{name:p.name,result:'positive',date:datetime("2021-11-10T16:40:32.142+0100")}
)
MERGE(p)-[t:TRANSACTED{timestamp:datetime("2021-11-09T18:40:32.142+0100")}]->(r:Re
staurants{postalcode:4172})
```

4. Conclusion & Improvements

We articulate the conclusion along with a number of improvements that we can apply to our model with respect to the information data: geolocation, travel information, weather information.

4.1 Geolocation

Location is the one of key information we use to connect input dataset pulled from different data sources.
The location can be referenced differently, for instance they can be coordinates, place names,or others and we need to ingest this data into the graph model in a reasoning way to make sure they are coherent.
Therefore, we need to build a module for processing geolocation. It will convert place names into geographic coordinates.

4.2 Travel information

As mentioned in the introduction paragraph, we could use the card transportation to monitor people when they use public transports or booking information when they take flights or trains to move to another city or country. We could also include car sharing data when people rent cars for short periods of time.

4.3 Weather information

Many research teams are investigating the correlation between COVID-19 disease and environmental variables such as temperature, humidity etc. A recent scientific report on nature.com (https://www.nature.com/articles/s41598-021-90300-9) claims a strong correlation based on the weather forecasting mechanism. Therefore, it's important to consider that kind of data in the model.