SMBUD - PROJECT WORK

Delivery #1

Academic year 2021/2022
Professor: Brambilla Marco

Group number: 4

Group members:
Elisa Mariani - 10632876
Pietro Guglielmo Moroni - 10625198
Giacomo Palù - 10682209
Francesco Panebianco - 10632465
Silvio Scanzi - 10622285

Politecnico di Milano

TABLE OF CONTENTS

1.Introduction

1.1. Specifics

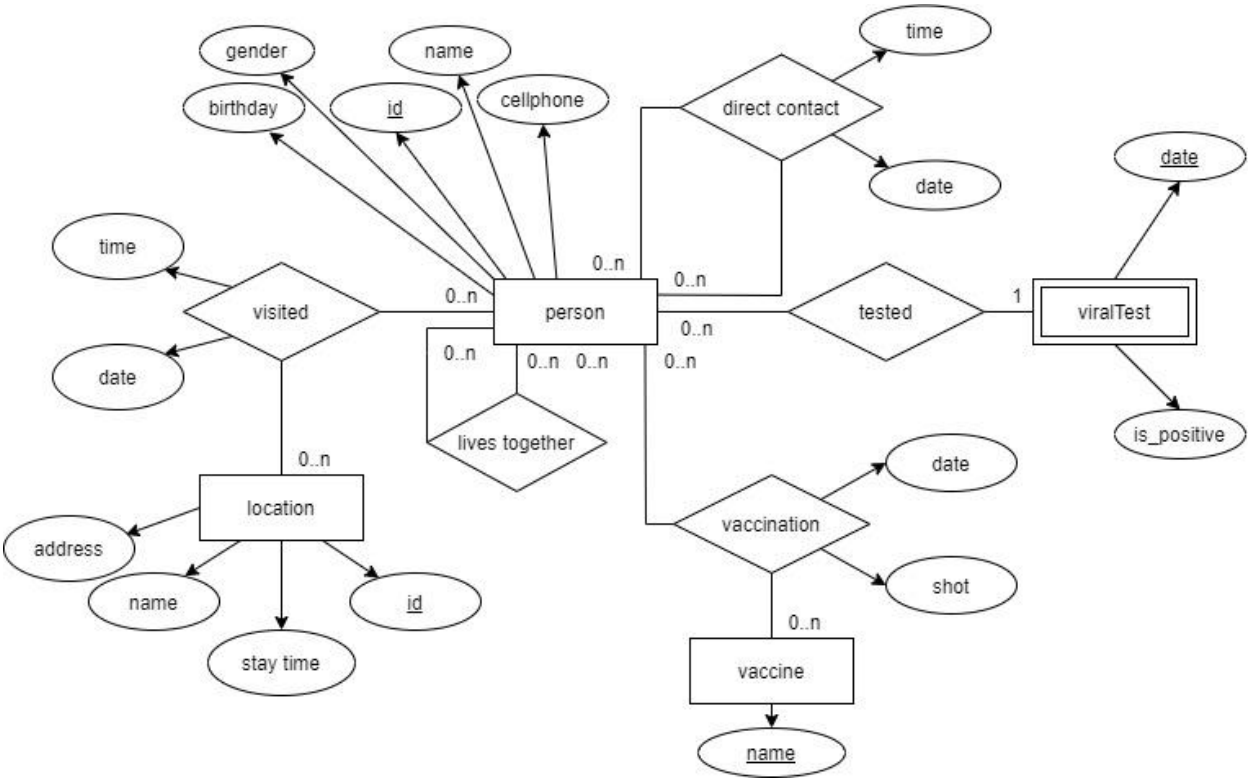
The database captures the state of the Covid-19 pandemic, focusing on contacts between people and the consequent contagions.
The database stores all the relevant information of each person, namely the fiscal code to identify him and, above all, the data regarding the Covid-19 pandemic; firstly if the person has been vaccinated, in which date and how many doses has he received; secondly the history of tests with their respective date in which they have been made and their result; lastly all the registered contagions of the person.
For contact tracing, the database needs to store all the possible contacts between each person, which can be of three different types. The first, obvious contact is "same family" contact: each person living in the same household will automatically be considered as direct contact. The second type of contact is the contact tracing apps and other possible devices which keep track of the people each individual has met in the past few days. The last type of contact is explicit data collection from establishments (such as restaurants, theaters, and so on).
To monitor the viral diffusion, the database needs to store all the relevant information of the contagion of the individuals, including date and place, when possible.

1.2. Hypotheses and implementation choices

Several assumptions have been made for the implementation of the database:

I. For each establishment that collects data, its name and its address are stored, as well as the average time spent in the establishment, to pinpoint the possible people who might have been in contact with an infected individual. In particular, the average time serves to evaluate with more precision possible contagions. For instance, if a place is visited by two people, the first in the morning and the latter in the evening, they should not be considered as people who came into contact.
II. Devices and tracing apps that provide data for the database collect only the date and time of the detected contacts.
III. The possible contagions happened for the visit of the same establishments can be extracted from the database through specific queries, which select anyone who has been in the same place and in the same time span as a positive person. Thus, no direct relationship from person to person is needed for this type of traced contacts.
IV. Positive people do not visit any place unless they take a negative viral test or at least 14 days have passed. Based on this assumption query number 3, in paragraph 4, has been modelled. In this query, when looking for places visited by positive people, we only consider who tested positive at the latest 5 days after the visit: we do not look for places visited by people in the days following the positive swab.
However, direct contacts traced by apps/devices remain possible also in the days after the positive swab.
V. The generated data concern the whole year '2021', even days to come.

2.ER model



The above ER diagram displays how the database is designed. This model describes the domain of the given problem, with specific concepts, attributes and relationships that we considered relevant and simple to manage. As the goal is to extract from the data useful information about the spread of Covid-19 among people, the main focus is on the entity Person and its interactions with the other aspects of the world. Every entity has a primary key, identified by the underlined attributes.
The entity ViralTest is weak, because every instance is dependant on the person who took the test, as well as on the date in which it was performed.
For the id attribute in entity Person, the fiscal code is used, as it is unique for each individual

3. Creation of the database

To populate the database, a simple yet effective approach was chosen: entities and relationships from the ER model were modeled in *Python*, along with procedures to randomly generate them. Each class that describes a node or edge type generally contains a `cypherCreate()` or a `cypherCreateRelationship()` method that returns the corresponding creation command in *Cypher* as a `string`. The `main.py` script, containing parameters for data size and features, coordinates the generation of a random population (composed of families, each with people sharing a surname), gathering places that can be visited, visits to said places, direct contacts between people and vaccination / testing event records. The complete data creation commands sequence is then generated and exported to a txt file.

4.Queries

Query #1. Find the people that tested 'positive' for the virus on 06-10-2020. Extract all the people that visited the same locations (at the same time) as them in the previous 10 days.Among them extract only those who had only 1 or 0 doses of vaccine at the moment of the contact.

The query:

```
MATCH  (swab:ViralTest{date:date("2020-10-06"),is_positive:true})←
    [:TESTED]-(infected_today:Person)-[v1:VISITED]→(location)←
    [v2:VISITED]-(potential_infected:Person)
OPTIONAL MATCH (potential_infected)-[vaxRel:VACCINATION]→()
WITH potential_infected,vaxRel
WHERE v1.date=v2.date
    AND ABS(duration.between(v2.time,v1.time).hours)<location.stay_time
    AND ABS(duration.between(v1.date,swab.date).days)<11
    AND v1.date<swab.date
    AND (NOT(vaxRel.shot>2))
    AND (NOT (vaxRel.shot=1 AND (potential_infected)-[:VACCINATION{shot:2}]→()))
    AND (NOT((vaxRel.dose=2)  AND (vaxRel.date<v1.date)))
RETURN  DISTINCT potential_infected.name
```

The outcome:

	potential_infected.name
1	"Adelino Alfano"
2	"Cirio Bonn"
3	"Ontario Alfano"

In this query, first of all we impose, to find the people infected on the day requested (*infected_people*): they must have a positive viral test on that day. In the same MATCH we look up also people who visited the same places as them (*potential_infected*). In the following OPTIONAL MATCH we define the relationship '*vaxRel*', which will be useful in the 'WHERE' to impose the constraints on the doses of vaccine. This clause has to be an 'optional' match because we want to include also the people without any dose. Lastly, in the WHERE clause we:

- impose that the '*infected_today*' and the '*potential_infected*' are in the same place and at the same time, computing the temporal interval between the two visits and comparing it with the average time of permanence of that place (exploiting durations functions)
- impose that the dates of the visits were at most 10 days before the positive swab
- check if the second dose of the vaccine was taken before or after the contact.

As long as the number of doses is concerned, this query has been tested in all the possible cases: people with only 1 dose (before and after the contact), 2 doses (both after the contact, both before and one before and one after) and more than 2(also in all the possible situations).

Query #2. Extract the average time of healing of people older than 60 years old (who have been infected at least once).

The query:

```
MATCH (swab1:ViralTest{is_positive:true})←[:TESTED] -
(person: Person)-[ : TESTED]→ (swab2:ViralTest{is_positive:false})
WHERE date.realtime().year-person.birthday.year>60 AND
swab1.date<swab2.date
WITH person.name AS Name,
MIN( duration. inDays( swab1.date, swab2.date).days) AS MIN_Time
RETURN AVG(MIN_Time)
```

The outcome:

	AVG(MIN_Time)
1	151.80555555555554

In the MATCH clause, we require that the people to look for must have a positive swab and a negative one. In the WHERE we then check that the positive test has been taken before the negative one and that the person is at least 60 years old (60 excluded). Lastly, we return for each person the name and the respective minimum time of healing(in the case of people infected multiple times).
In this last particular case (of multiple infections and consequent healing), the time of healing considered is in fact the lowest one.
Since the data in the dataset are created randomly, we can accept that the outcome of this query is unrealistic.

Query #3. Extract the 10 places that in 2021 were most visited by people while they were probably positive (after maximum 5 days they tested positive).

The query:

```
MATCH (positive_test:ViralTest{is_positive:true})←[:TESTED]-
        (person:Person)-[visit_rel:VISITED]→(place:Location)
WHERE visit_rel.date.year=2021 AND positive_test.date>visit_rel.date
        AND duration.inDays(positive_test.date,visit_rel.date).days<6
WITH place,COUNT( DISTINCT visit_rel) AS number_of_visits_by_positive
ORDER BY number_of_visits_by_positive DESC
LIMIT 5
RETURN place.name+","+place.address AS Places_most_visited_by_positive_people
```

The outcome:

	Places_most_visited_by_positive_people
1	"Ristorante Anklin,Via IGINIO UGO TARCHETTI, 88"
2	"Pizzeria Jianu,Viale ABRUZZI, 8"
3	"Dentista Michilli,Via Della SPIGA, 47"
4	"Ristorante Baiano,Via FARINE, 9"
5	"Scuola Tronconi,Via BROLETTO, 16"

In this query, with the MATCH clause, we look for people who both got tested positive and visited some places. Then we impose that the date of the viral test should not be more than 5 days after the visit and that both dates should be in the year '2021'. In lines 5-6 we count the visits with the previous requirements that each place had and we order all the places based on that (in descending order). With the LIMIT clause than we select only the first five places.

Query #4. Extract the names of all the people that in the last 5 days from now came in contact (according to tracing app/devices or for cohabitation) with currently infected people. Extract also the names of the people that came in contact with the previous ones (again with contact tracing apps/devices or for cohabitation) in the same 5 days (two-steps connections).

The query:

```
1  MATCH (nowpositive: Person)-[ : TESTED] ->(swab:ViralTest),
2  (nowpositive)-[dir1: DIRECT_CONTACT] ->(withApp: Person),
3  (withApp)-[dir2: DIRECT_CONTACT] ->(withApp_withApp: Person),
4  (withApp)-[dir3:LIVES_TOGETHER] ->(withApp_withFamily:Person),
5  (nowpositive)-[fam1:LIVES_TOGETHER]-> (withFamily:Person),
6  (withFamily)-[fam2:DIRECT_CONTACT] -> (withFamily_withApp: Person)
7   WITH dir1,dir2, dir3, fam1, fam2, withApp, withApp_withApp, withApp_withFamily,
8  withFamily, withFamily_withApp, nowpositive AS P, swab AS S
9   ORDER BY S.date DESC
10  WHERE
11     (ABS(duration.inDays(dir1.date,date.realtime()).days)<5 AND dir1.date<date.realtime())
12   OR
13     (ABS(duration.inDays(dir1.date,date.realtime()).days)<5 AND
14      dir1.date<date.realtime() AND ((ABS(duration.inDays(date.realtime(), dir2.date).days)<5
15      AND withApp_withApp<>P AND dir2.date<date.realtime()) OR
16      (ABS(duration.inDays(date.realtime(), dir3.date).days)<5 AND withApp_withFamily<>P
17      AND dir3.date<date.realtime())))
18   OR
19     (ABS(duration.inDays(date.realtime(), fam1.date).days)<5 AND fam1.date<date.realtime())
20   OR
21     ((ABS(duration.inDays(date.realtime(), fam1.date).days)<5) AND fam1.date<date.realtime()
22     AND (ABS(duration. inDays(date.realtime(), fam2.date).days)<5 AND fam2.date<date.realtime()
23     AND withFamily_withApp<>P)) AND swab.date<date.realtime()
24  WITH dir1,dir2, dir3, fam1, fam2, withApp, withApp_withApp, withApp_withFamily,
25 withFamily, withFamily_withApp, P, head(COLLECT(S)) AS lastSwab
26  WHERE lastSwab.is_positive=true
27  RETURN COLLECT (DISTINCT withApp.name) AS OneStepConnection_withApp,
28  COLLECT(DISTINCT withApp_withApp.name)AS TwoStepConnection_withApp_withApp,
29  COLLECT(DISTINCT withApp_withFamily.name) AS TwoStepConnection_withApp_withFamily,
30  COLLECT (DISTINCT withFamily.name) AS OneStepConnection_withFamily,
31 COLLECT(DISTINCT withFamily_withApp.name) AS TwoStepConnection_withFamily_withApp
```

The outcome:

"OneStepConnection_withApp"	"TwoStepConnection_withApp_withApp"
["Fulvia Marchesini","Adeodata Milani","Simonino Kosek","Claro Callegarini","Regina Bentivegna","Fatima Leonetti","Biagio Vella","Bonomo Pezzuco"]	["Fatima Leonetti","Cleto Bentivegna","Gervasa Di","Gualfredo Sbreni","Leopardi Poni","Ponziano Bianco","Ervin Cichon","Emera Popovici","Giauario Colli","Claro Callegarini","Filippino Bazzocchi","Olivieri Patwary","Frenesia Sbreni","Tersiglia Mena","Bonomo Pezzuco","Maira Boes","Adeodata Milani","Nelita Assal","Fierino Milani","Annachiara Raqabi","Feliziana Ruggeri","Biagio Vella","Elleno Grande","Manilla Haryadi","Orvieto Paglia","Giovammaria Panchetti","Matrona Pacini","Fulvia Marchesini","Ottima Morselli","Fortura Assal","Terzilia Pastore","Regina Bentivegna","Odette Alfano","Fiora Balta","Simonino Kosek","Ermido Baragliu","Clora Corrias"]

"TwoStepConnection_withApp_withFamily"	"OneStepConnection_withFamily"
["Ervin Marchesini","Ornelia Milani","Fierino Milani","Idreno Milani","Baldovina Kosek","Dusola Kosek","Pubblio Kosek","Fiordalice Callegarini","Francolina Callegarini","Flera Callegarini","Cleto Bentivegna","Weber Leonetti","Cunegonda Vella","Adeodato Pezzuco","Drusilla Pezzuco"]	["Weber Leonetti","Fiordalice Callegarini","Francolina Callegarini","Flera Callegarini","Adeodato Pezzuco","Drusilla Pezzuco","Ornelia Milani","Fierino Milani","Idreno Milani","Cunegonda Vella","Ervin Marchesini","Cleto Bentivegna","Baldovina Kosek","Dusola Kosek","Pubblio Kosek"]

"TwoStepConnection_withFamily_withApp"
["Crescenzo Ianne","Fortura Assal","Rinello Ianne","Leopardi Poni","Antonetta Grande","Clora Corrias","Cristofaro Casali","Oliverio Mariani","Norina Bonn","Davina Pello'","Leonilde Pello'","Angiolina Mamone","Donillo Strazzanti","Ervin Marchesini","Filadelfia Colli","Ennio Pastore","Rustico Forcina","Polonia Piro","Nannino Mariani","Severo Maroncelli","Ruffino Sassi","Regina Bentivegna","Apollonia Lelli","Irenea Emumwen","Bino Mariani","Cenerina Ruggeri","Abramo Sassi","Erveno Linzi","Fenisio Zepponi","Matrona Pacini","Elmina Guerra","Adina Bazzocchi","Italiano Montanari","Ornelia Milani","Katia Caraballo","Torquata Testa","Adeodato Pezzuco","Censo Ranfi","Adalindo Assal","Ontario Alfano","Sabato Strazzanti","Fondina La","Gesuina Rocchi","Telma Garofalo","Fulvia Marchesini","Frenesia Sbreni","Fara Di","Filomeno Ossati","Severa Carullo","Ita Piro","Falerio Panchetti","Gilberto Corrias","Edilia Cellini","Lalla Pastore","Terzino Alfano","Angiolo La","Lorino Popovici","Ostillo Forcina"]

In the first MATCH clause, we look up for people who took at least one swab and who had a direct contact (detected by tracing apps/devices).
The following 'optional matches' instead are for including people who came in contact with the previous ones (second step connections).
Then, we order the swabs by date from the newest to the oldest one. This is necessary to select the most recent one and impose its positivity (in the last WHERE clause).
In the middle 'WHERE' instead we make sure that all the contacts are up to 5 days old.In our case the checks of the type "*dateX<date.realtime()*" are necessary because the DB includes data concerning all the current year.

Query #5. Extract, among all the people not vaccinated, the percentage of people younger than 25 years old.

The query:

```
MATCH (nonVaccinated_young:Person),(nonVaccinated_tot:Person)
WHERE date.realtime().year-nonVaccinated_young.birthday.year⩽25
    AND NOT (nonVaccinated_young)-[:VACCINATION]→()
    AND NOT (nonVaccinated_tot)-[:VACCINATION]→()
WITH COUNT(DISTINCT nonVaccinated_tot) AS total_number,
     COUNT(DISTINCT nonVaccinated_young) AS young_number
WHERE total_number<>0
RETURN (young_number*100/total_number) AS Percentage
```

The outcome:

	Percentage
1	19

In this query, in the first "WHERE" we check the age of the people we want to find the percentage of and we impose that both them and the *'nonVaccinated_tot'* must not have any vaccination. Then we count the two amounts of people to place them as the numerator and denominator in the calculation of the percentage (imposing the constraint denominator>0 in the 'WITH' clause).

Query #6. Extract the percentage of people who stopped using tracking apps/devices(no direct contacts detected) after they got vaccinated with the second dose.

The query:

```
MATCH ()←[:VACCINATION{shot:2}]-(total_users:Person)-[:DIRECT_CONTACT]→()
MATCH ()←[vaxRel:VACCINATION{shot:2}]-(lost_users:Person)-
[directRel:DIRECT_CONTACT]→()
WHERE NOT directRel.date>vaxRel.date
WITH COUNT(DISTINCT total_users) AS number_total, COUNT(DISTINCT
lost_users) AS number_lost
WHERE number_total<>0
RETURN (number_lost*100/number_total) AS Percentage_lost_users,
number_total, number_lost
```

The outcome:

	Percentage_lost_users
1	88

In this query, initially, we look up for "total_users" and "lost_users", that both have to have at least a vaccine and a direct contact. In the 'WHERE' clause we specify that the '*lost_users'* are the ones that have no direct connections after the date in which they had the second dose of vaccine. Then, we proceeded to calculate the wanted percentage as in query #5.

5. Commands

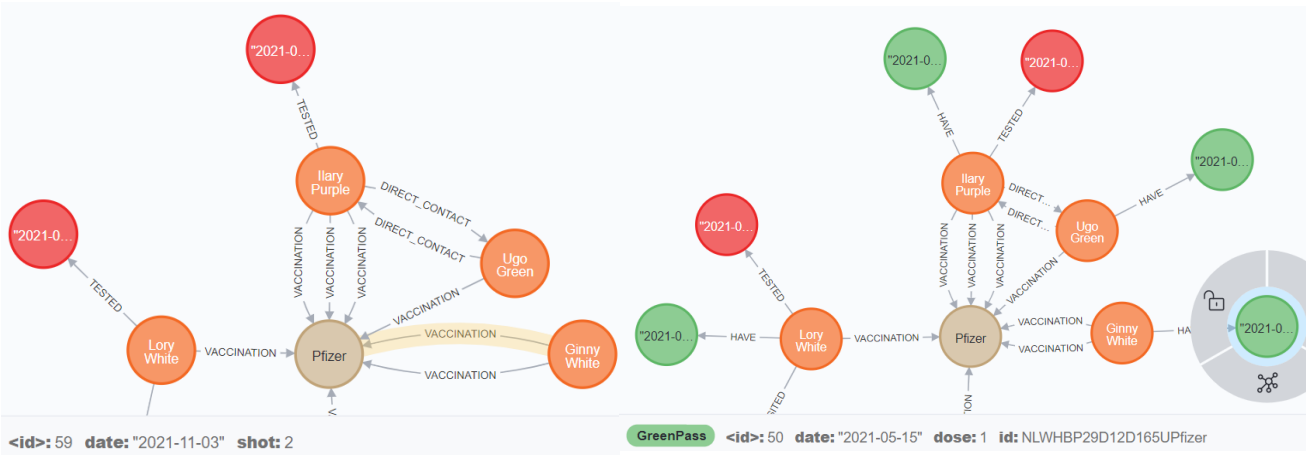
The small graphical examples presented in this paragraph were performed on a smaller DB in order to facilitate the comprehension of the images. However, the commands have been tested and work just as well on the final dataset, even though the command#2 takes a long time to finish in that case.

Command #1. For each person with a vaccine (1,2 or more doses) from at least 14 days, create a 'GreenPass' with an identification code (composed of the fiscal code and name of the vaccine concatenated), the number of the dose, and the release date (14 days after the vaccine). Create only Green Passes related to the last dose of each person.

```
MATCH (vaccinated:Person)-[vaxRel:VACCINATION]→(vaccine:Vaccine)
WITH vaccine,vaccinated,vaxRel
ORDER BY vaxRel.date DESC
WHERE duration.inDays(vaxRel.date,date.realtime()).days ⩾ 14
WITH head(COLLECT(vaxRel)) AS last_vaccine_rel,vaccinated,vaccine
MERGE(greenpass:GreenPass{id:vaccinated.id+vaccine.name,
    date:last_vaccine_rel.date+duration("P14D"),dose:last_vaccine_rel.shot})
    ←[:HAVE]-(vaccinated)
```

First of all, we impose that the person must have at least 1 dose of vaccine. Then we order the doses of vaccine from the most recent one and we check that the vaccinations must be at least 14 days old. Then we take the most recent vaccination for each person to generate the green pass with the requested information and we connect this new entity "*GreenPass*" to the vaccinated person.

A small example: before and after the command.



If we look at the node of '*Ginny White*' we can see that she was vaccinated with the second dose only on 03-11-2021. Since no more than 14 days have passed yet(from 03-11-2021), she will have the Green Pass for the first dose, as we can see in the second image.

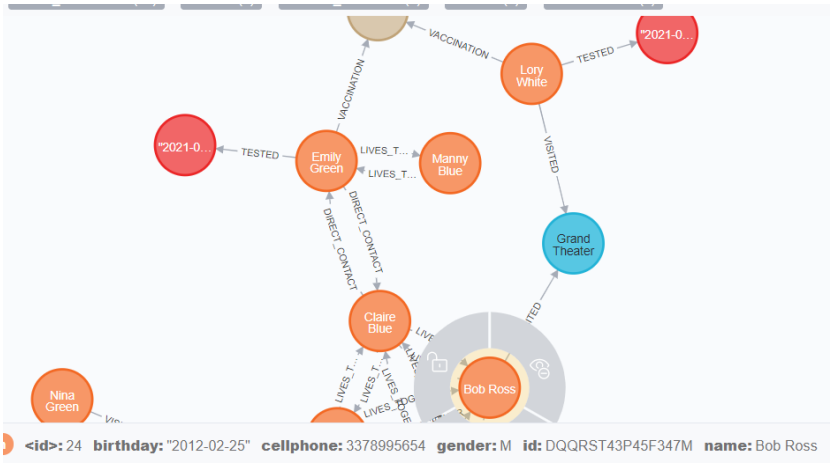
Command #2. Add the attribute "*possible_contagion*" to each person that has been at risk for infection. The attribute specifies the date and the time (when possible also the place) of the possible contagion.

```
MATCH (p:Person)
SET p.possible_contagion=[]
```

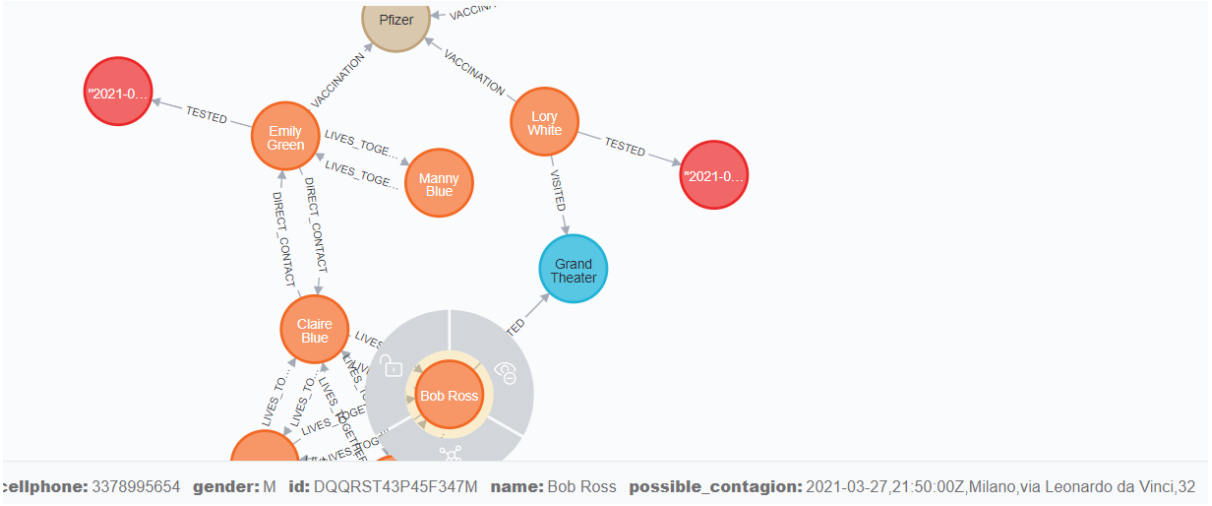
```
MATCH (potential_infected1:Person)-[potential_visited:VISITED]→(location)
    ←[infected_visited:VISITED]-(infected1:Person)-[:TESTED]→(viralTest:ViralTest{is_positive:true})
MATCH (potential_infected2:Person)-[direct:DIRECT_CONTACT]→(infected2:Person)
    -[:TESTED]→(viralTest2:ViralTest{is_positive:true})
WHERE duration.inDays(viralTest.date,infected_visited.date).days⩽14 AND
    viralTest.date>infected_visited.date
    AND potential_visited.date=infected_visited.date
    AND ABS(duration.between(potential_visited.time,infected_visited.time).hours)<location.stay_time
    OR (duration.inDays(viralTest2.date,direct.date).days⩽14 AND viralTest2.date>direct.date)
WITH potential_infected1,potential_infected2,
(toString(potential_visited.date)+","+toString(potential_visited.time)+","+location.address) AS
    potentialDateAndPlace,(toString(direct.date)+","+toString(direct.time)) AS potentialDateAndTime

WITH COLLECT(DISTINCT potentialDateAndPlace) ASpotentialDateAndPlace2,potential_infected1,
    potential_infected2,potentialDateAndTime,
    [x IN potential_infected1.possible_contagion WHERE x<>potentialDateAndPlace] AS modifiedCollection1
SET potential_infected1.possible_contagion=modifiedCollection1+potentialDateAndPlace2

WITH COLLECT(DISTINCT potentialDateAndTime) AS potentialDateAndTime2,potential_infected2,
    [x IN potential_infected2.possible_contagion WHERE x<>potentialDateAndTime] AS modifiedCollection2
SET potential_infected2.possible_contagion=modifiedCollection2+potentialDateAndTime2
```

In the first command, we select all people and create an attribute '*possible_contagion',* initialized as an empty list.
In the second 'MATCH' clause we select both people that have been in the same place as people with a positive test ('*potential_infected1'*) and people that have been in direct contact with people with a positive test ('potential_infected2').
In the 'WHERE', we check that the distance between the date of the visit and the date of the positive swab must not be more than 14 days. The same thing for the date of the *'direct contact*'. We also check that the visit must have happened on the same day and during the same hours (checking with the average time of permanence of the place, '*location.stay_time'*).
In the following 'WITH' clauses, we create the strings to insert in the attributes *"possible_contagion*". To do that, we concatenate the strings of the date, the time, and location (the latter only if the contact happened in registered places). Then, we collect the strings created with the 'DISTINCT' clause to avoid duplicates. For the same reason, to avoid adding an already existing string to the collection, we created a filter (line 16,20) able to remove that element (if present) from the previous collection and save this modified one as 'ModifiedCollection'. With the final 'SETS', we add the string created to the modified one(which in fact will be modified only if already containing the string of possible contagion).
This last step was necessary in the case of multiple people tested positive who have been in the same places, to avoid having multiple strings with the same address and the same time in the attribute *"possible_contagion*".
In particular, we had to differentiate the two final "WITH" and "SET" between the two cases because otherwise, if a Person had both a dangerous *'direct contact'* and a dangerous '*visit',* the attribute '*possible_contagion*' would have been overwritten, taking into consideration only one of the two scenarios.

A small example: before and after the two commands.



Lory White and Bob Ross visited the Grand Theater on the same day, in the same hours. After that Lory White tested positive for the virus.
As expected, after the two commands, in the node of Bob Ross appears the attribute "possible_contagion" with the address of the Grand Theater.

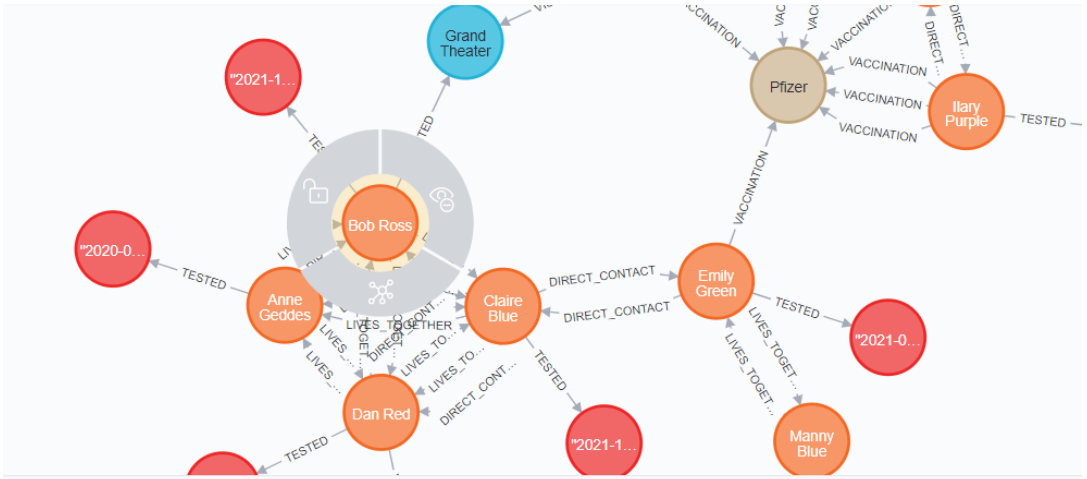


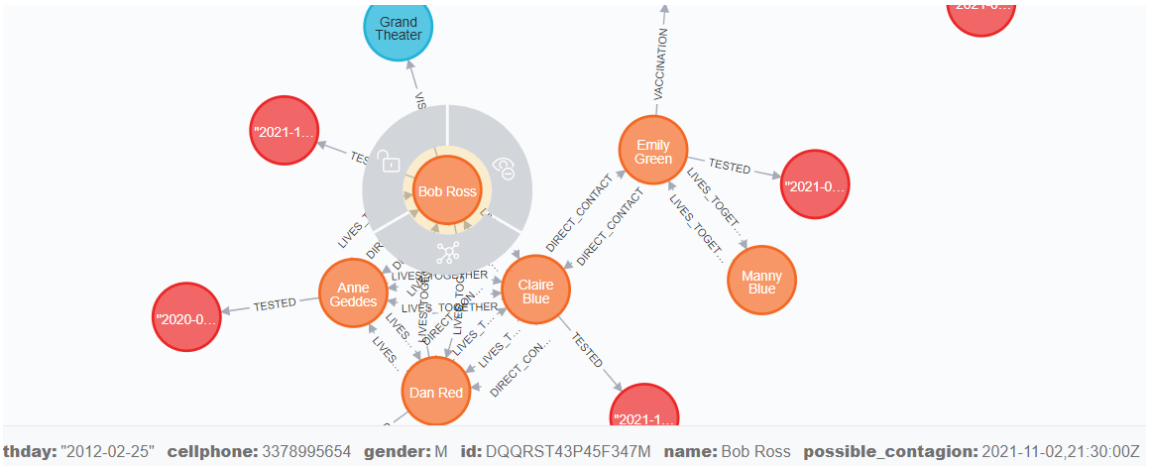
Command #3. For each person that tested negative for the virus today, delete data on possible infections (strings of the Person's attribute *'possible_contagion')* dating back more than 10 days.

```
MATCH (p:Person)-[:TESTED]→(viralTest:ViralTest{is_positive:false})
WHERE viralTest.date=date.realtime()
WITH p,[x IN p.possible_contagion WHERE
        ABS(duration.inDays(date.realtime(),date(left(x,10))).days)<10] AS modifiedList
SET p.possible_contagion=modifiedList
```

This command selects all the people with a negative test taken today. Then, for each string of the attribute "*possible_contagion"* (that is an array of strings) selects the first 10 characters(*'left(...)'*) and convert them into a date(*'date(..)'*) and leaves in the array only the strings containing dates 10 days old at most. Lastly, replaces the old array with the new one in the variable "*possible_contagion"*.

A small example: before and after the command.





As expected, the command deleted only the possible contagion dating back to 2021-03-27 because it's more than 10 days ago. Instead, left the second one because it's about a contagion that happened on 02-11-2021 (10 days have not passed yet).

6. Application

The application has been developed to show how this database could be used in a real-world scenario. The application has two main purposes, firstly to show the data in a meaningful way (show how many contagions/negative swabs have been reported in a day/month) in order to monitor efficiently the covid-19 pandemic. Secondly, the application allows to update the database, inserting new data about swabs. This functionality automatically deletes all possible contagions of the person (in a reasonable time frame, according to the date of the test) if the swab is negative, and informs all the people who have been in contact with the positive person if the swab is positive.
This simple application shows that indeed the database has been built according to the requirements and could effectively be used in a real-world scenario.
Further information is provided in the user guide.