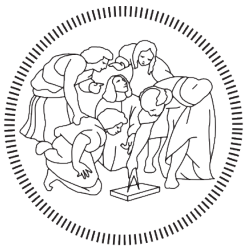
Academic Year 2020/2021



POLITECNICO
MILANO 1863

Computer Science and Engineering

System and Methods for Big and Unstructured Data

Project Report - Neo4j

Matteo Falzi - 10638723
Fabio La Manna - 10620549
Filippo Manzardo - 10864201
Paolo Marzolo - 10668259
Matteo Regge - 10619213

Prof. Marco Brambilla

November 14, 2021

1 Introduction

The development of applications for tracing people contacts have seen an increasing interest due to Covid pandemic. The aim of these apps is to record paths and meetings of people in order to know their contacts and so to contain the spread of the virus. Recording paths means knowing where a person has been in the previous days, which places he visited and how much time he spent in there. This is important because in case of positive infection, these places can be advised in order to contact all the other people who were in the same place in the previous days. Recording meetings means knowing which people were directly in contact and how much time they spent together. This is helpful because in case of positive infection, all the people who were in contact with the infected can be found and then proceeded to be tested. The project consists of modelling a database to exploit these contact tracing applications using a NoSQL DB called Neo4J.

1.1 Delivery specification

The goal of the project is: designing, storing and using a graph structure in a NoSQL DB for supporting a contact tracing application. The activities of the users can be: visiting some places or directed contacts. It has to be discerned the family group from the other people because family members are in contact by default. Every person must have some data: vaccines, tests and contagion information. In addition it has to be provided some queries and commands to retrieve information from the database. Obviously in order to do this, it will be needed a sample dataset that imitates a real Covid situation.
All the work done can be found at this link: GitHub Repo

1.2 Assumptions

Users: Users are over the age of 18. We have plenty of information about their addresses and birth. We have access to their Medical Records, tracking only information related to COVID, for example COVID Tests, COVID Vaccines, Health Status and general Health Risk. Due to the access of COVID Tests, we know if an user is Positive to the virus. The attribute "Last Confirm" indicates the date of the latest test in which the person could be resulted positive or negative. So the contagion date is implicit in this attribute because a person is revealed positive only when he obtains a positive result from the latest test.

Places: The dataset containing places was retrieved from GeoNames and it has been preprocessed for better use. We distinguish places that have rooms. To simplify, we assumed that every place exist and can be reached by an user.

Visits: Every visit an user makes it's tracked by a third-party systems. We assume that in all the places, people are asked for their personal data and it is also registered the duration and the date of the visit.

Contacts: We consider a contact between the user and another person in a specific date and we always know the duration of the meeting. Every contact is mutual, meaning if P1

contacts P2, P2 contacts P1. The meeting between people can be tracked via the application on the smartphone but also on other devices like: bracers and smartwatches (we assume every person as at least a smartphone).

Personal data: We consider that a person may or may not have a vaccine/test. Also, we limit the vaccines into 4 categories: Pfizer, Astrazeneca, Johnson&Johnson, Moderna. Any user cannot have more then one Vaccine/Test per day.

Lives: We consider family members as people who live in the same house. We modelled this with a relation "lives" between people who live in the same house.

2 Relational Model

To approach the problem, we developed an ER Model based of 3 main entities: Person, Medical Record and Place. Following the image of the ER diagram and the descriptions of the classes and relations.

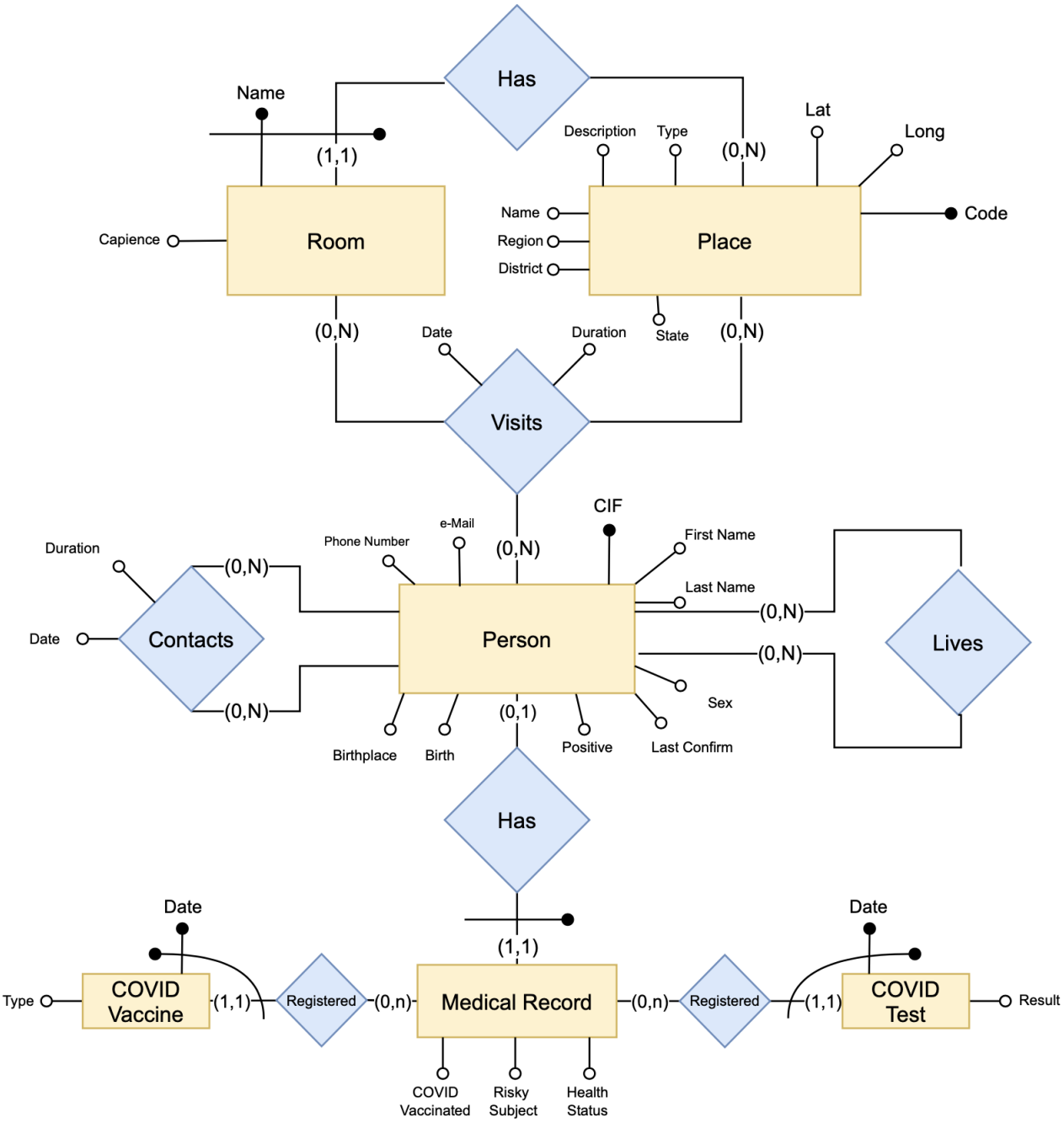


Figure 1: Entity-Relation Model

2.1 Logical Model

```
####### Entities #######
Person(CIF,First_Name,Last_Name,Sex,Birth,Birthplace,Phone_Number,..
       ..e-Mail,Positive,Last_Confirm)
Place(Code,Name,Lat,Long,CAP,State,Region,District,Type,Type_Desc)
Rooms(Code,Name,Capience)
Medical_Record(CIF,Covid_Vaccinated,Risky_Subject,Health_Status)
Covid_Tests(CIF,Date,Results)
Covid_Vaccines(CIF,Date,Type)

####### Relations #######
Contacts(CIF,CIF,Date,Duration)
Lives(CIF,CIF)
Visits(CIF,Code,Name,Date,Duration)
```

3 Neo4j Implementation

To generate entities and relations, we coded a python script called "generate_dataset.py", in the latest DB there are:

- 8000 People
- 30000 Contacts
- 500 Places
- 500 Rooms
- 20000 Visits

For a total of 38904 nodes and 88110 relations. These numbers can be modified through the variables found in the first lines of the script.
Once that the script has been executed, 9 output CSV files can be found in scripts/output. This file are later imported into Neo4j's database via some load queries listed in "scripts/load_queries.txt".

3.1 Queries

Now we list some queries we wrote to inspect our database for useful information about COVID.

Query n. 1

Find all the contacts a Positive Person made during the last 5 days, returning Names, Dates and Days Passed

---

```
MATCH (p1:Person)-[c:CONTACTS]-(p2:Person)
WITH p1, p2, p1.Last_Confirm as positiveness_date, c.Date as contact_date
WHERE positiveness_date > contact_date and
    positiveness_date < contact_date + duration(days: 5) and
    p1.Positive = True
RETURN DISTINCT p1.First_Name + " " + p1.Last_Name as Positive,
    p2.First_Name + " " + p2.Last_Name as Contacted, positiveness_date, contact_date,
    duration.inDays(contact_date, positiveness_date).days  as days_passed
    LIMIT 25
```

"Positive"	"Contacted"	"positiveness_date"	"contact_date"	"days_passed"
"Jennifer Floyd"	"Elizabeth Bozwell"	"2021-10-24"	"2021-10-21"	3
"Joan Elkins"	"Justin Ferris"	"2021-09-30"	"2021-09-26"	4
"Charlotte Majeski"	"Gary Cody"	"2021-10-01"	"2021-09-28"	3
"Mary Zahar"	"Kathleen Ward"	"2021-10-01"	"2021-09-30"	1
"Cathy Czelusniak"	"Rosa Reynolds"	"2021-10-31"	"2021-10-30"	1
"Wanda Schrader"	"Erick Matyas"	"2021-10-11"	"2021-10-08"	3
"Reginald Winkler"	"Abraham Mitchell"	"2021-10-26"	"2021-10-25"	1

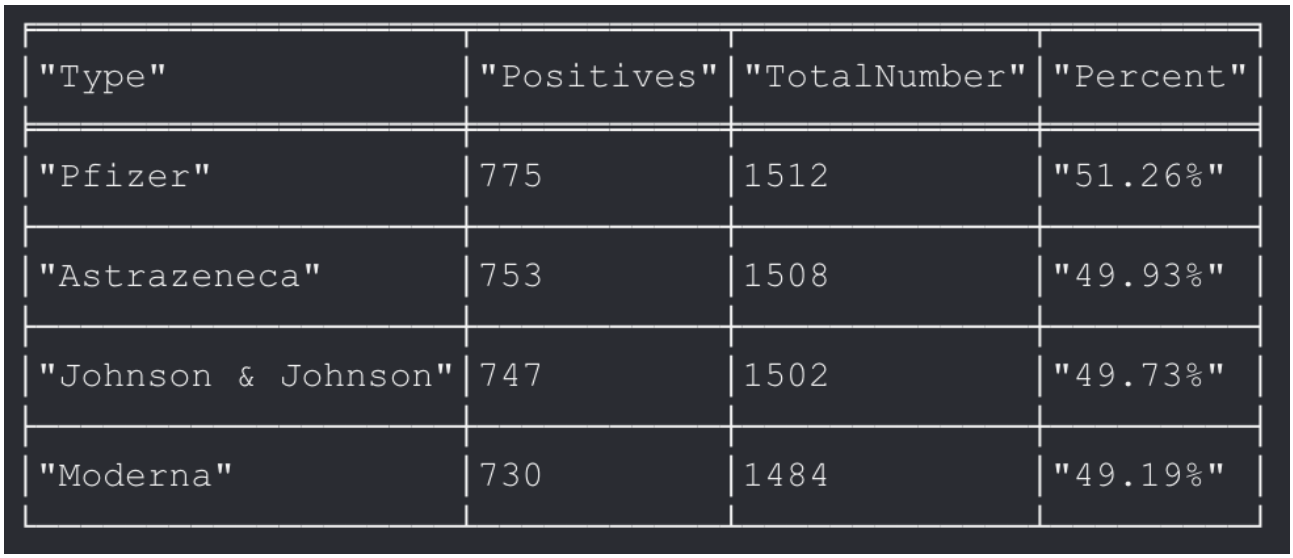
Query n. 2

Find how many positive vaccinated people have a determined type of Vaccine, returning Positive to Total ratio

---

```
MATCH (m)-[:REGISTERED]->(c:Covid_Vaccines)
WITH c.Type as Type, count(*) as TotalNumber
ORDER BY c.Type
```

```
MATCH (p:Person)-[:HAS]-(m:Medical_Record),
    (m)-[:REGISTERED]->(c:Covid_Vaccines)
WHERE p.Positive = True and Type = c.Type
RETURN  Type, count(*) as Positives, TotalNumber,
round(ToFloat(count(*)) / ToFloat(TotalNumber)*100, 2) + "%" as Percent
ORDER BY Percent DESC
```



"Type"	"Positives"	"TotalNumber"	"Percent"
"Pfizer"	775	1512	"51.26%"
"Astrazeneca"	753	1508	"49.93%"
"Johnson & Johnson"	747	1502	"49.73%"
"Moderna"	730	1484	"49.19%"

Query n. 3

Find which places and rooms have been visited by positive person in the last 5 days for a duration time longer than 50 minutes

---

```
MATCH (p1:Person)-[v1:VISITS]->(pl:Place)
WHERE p1.Last_Confirm > v1.Date and
    p1.Last_Confirm < v1.Date + duration(days:5)
    and
    p1.Positive = True and
    v1.Duration > time(hour:0,minute:50)
RETURN DISTINCT p1.CIF as CIF, p1.First_Name as Name, pl.Code as Place,
pl.Room as Room, v1.Date as Date,p1.Last_Confirm as Positiveness_Date
ORDER BY p1.CIF
UNION
MATCH (p1:Person)-[v1:VISITS]->(r:Rooms)
```

```
WHERE p1.Last_Confirm > v1.Date and
    p1.Last_Confirm < v1.Date + duration(days:5)
    and
    p1.Positive = True and
    v1.Duration > time(hour:0,minute:50)
RETURN DISTINCT p1.CIF as CIF, p1.First_Name as Name, r.Code as Place,
r.Name as Room, v1.Date as Date, p1.Last_Confirm as Positiveness_Date
ORDER BY p1.CIF
```

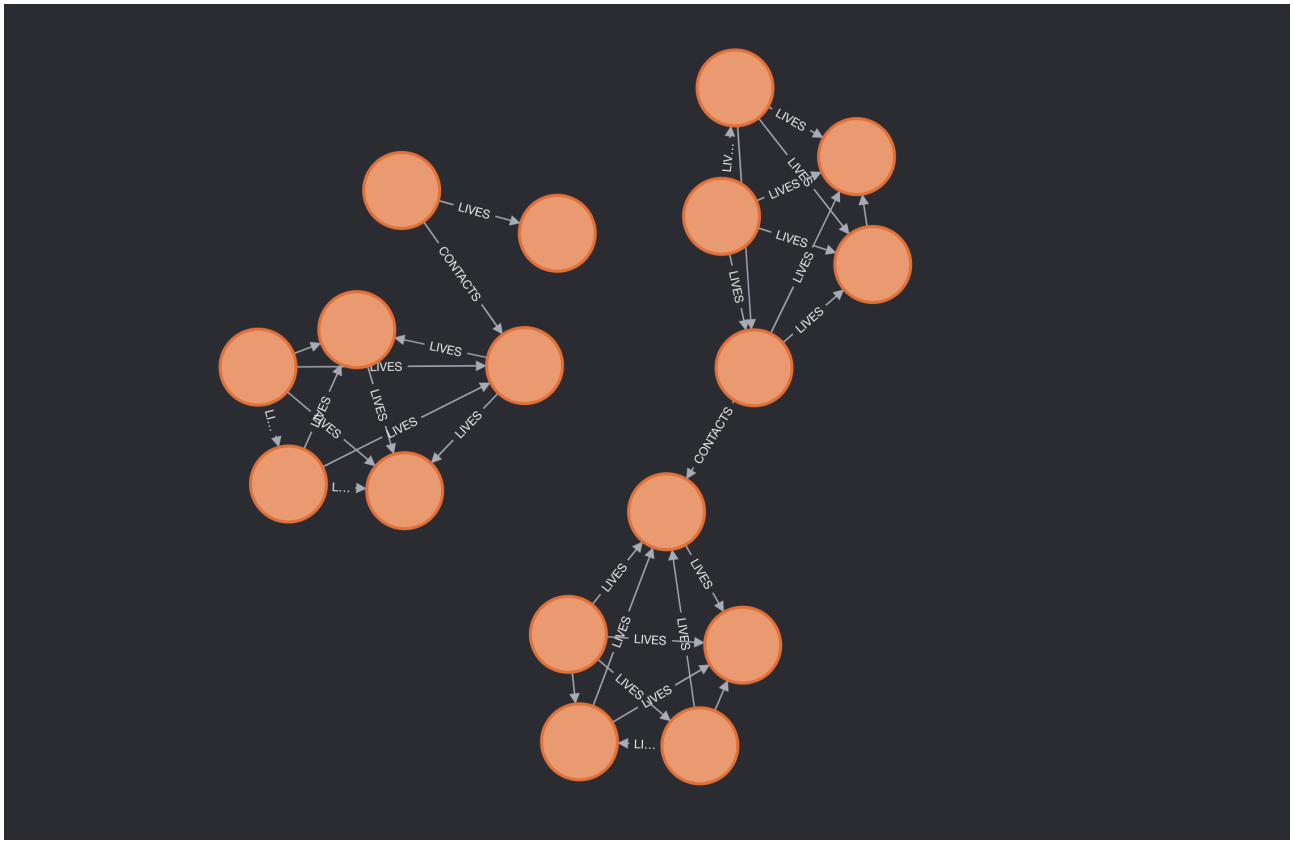
"CIF"	"Name"	"Place"	"Room"	"Date"	"Positiveness_Date"
"BRSMDR73R50H010F"	"Mildred"	6618632	null	"2021-10-07"	"2021-10-10"
"BRYPTR78E18I439A"	"Peter"	8985609	null	"2021-10-13"	"2021-10-16"
"CHLNGL76A52L186T"	"Angela"	10338853	null	"2021-11-05"	"2021-11-08"
"DVRVNE86S24E113L"	"Evan"	3219375	null	"2021-11-08"	"2021-11-09"
"DYEBNY83R43H338G"	"Ebony"	3173137	null	"2021-11-07"	"2021-11-11"
"FGNRTR00R16C252Z"	"Arturo"	3180654	null	"2021-10-05"	"2021-10-08"
"FSCHHR90C61E008D"	"Heather"	11818436	null	"2021-10-05"	"2021-10-09"
"GRDPTR53R31C236H"	"Peter"	11128068	null	"2021-11-05"	"2021-11-08"

Query n. 4

Track all the people who were in contact with a positive in the last 15 days from his confirmation and find all their familiars

---

```
MATCH (p1:PersonPositive:true)-[c:CONTACTS]-(p2:Person)
WHERE p1.Last_Confirm >= c.Date and p1.Last_Confirm <= c.Date + duration(days: 15)
MATCH (p2)-[:LIVES*]-(p4:Person), (p1)-[:LIVES*]-(p3:Person)
RETURN DISTINCT p1, p2, p3, p4
```



Query n. 5

Track all the people who were in the same place of a positive in the last 15 days from his confirmation

---

```
MATCH (p1:PersonPositive:true)-[v1:VISITS]->(pl:Place)<-[v2:VISITS]-(p2:Person)
WHERE p1.Last_Confirm >= v1.Date and p1.Last_Confirm >= v2.Date and
p1.Last_Confirm <= v1.Date + duration(days: 15) and
```

```
p1.Last_Confirm <= v2.Date + duration(days: 15)
RETURN DISTINCT p1.First_Name + " " + p1.Last_Name as Positive,
p2.First_Name + " " + p2.Last_Name as Contacted, p1.Name as Place, p1.Last_Confirm,
v1.Date as Positive_Visit, v2.Date as Contact_Visit
```

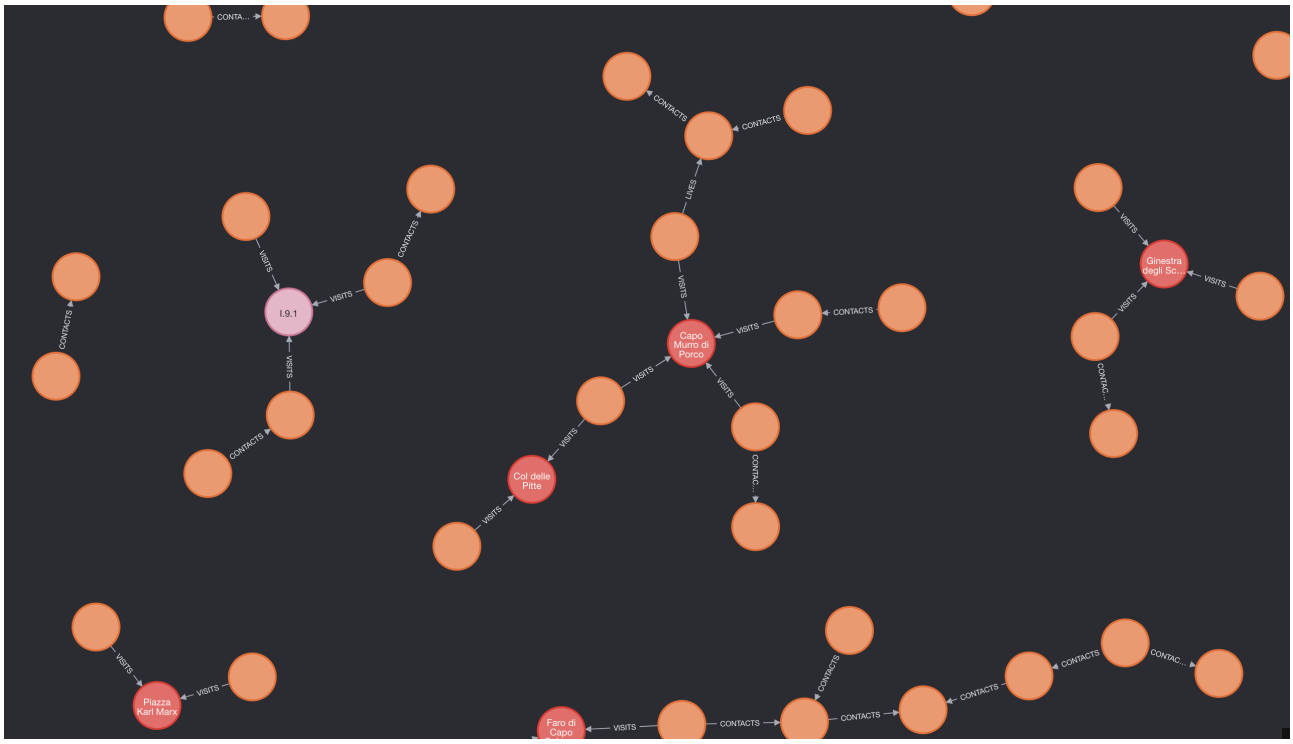
"Positive"	"Contacted"	"Place"	"p1.Last_Confirm"	"Positive_Visit"	"Contact_Visit"
"Arnold Concepcion"	"Ronald Chauvin"	"Monte Cupoli"	"2021-10-22"	"2021-10-11"	"2021-10-13"
"Arnold Concepcion"	"Lisa Sutton"	"Monte Cupoli"	"2021-10-22"	"2021-10-11"	"2021-10-13"
"Elizabeth Pantoja"	"Dorothea Santiago"	"Comazzo"	"2021-11-03"	"2021-10-24"	"2021-10-27"
"David Polczynski"	"Berniece Jones"	"Pienza"	"2021-10-20"	"2021-10-18"	"2021-10-05"
"David Polczynski"	"George Keene"	"Pienza"	"2021-10-20"	"2021-10-18"	"2021-10-09"
"David Polczynski"	"Emily Bell"	"Pienza"	"2021-10-20"	"2021-10-18"	"2021-10-05"
"Peter Bruyere"	"James Croom"	"Via Malignan"	"2021-10-16"	"2021-10-13"	"2021-10-06"
"Peter Bruyere"	"Jennifer Miller"	"Via Malignan"	"2021-10-16"	"2021-10-13"	"2021-10-16"
"Camille Carpenter"	"Raymond Rains"	"Monastier di Treviso"	"2021-11-07"	"2021-10-29"	"2021-11-02"
"Grace Markham"	"Michael Vaillancourt"	"Cima di Costabella"	"2021-10-01"	"2021-09-16"	"2021-09-30"
"Carmen Trump"	"Chad Vice"	"Piona"	"2021-11-07"	"2021-10-27"	"2021-10-28"

Query n. 6

Track all the people who were in contact, also indirectly, with a positive or with one of his contacts in the last 15 days of his confirmation.
The possible infected are risky subjects and they don't have a vaccine

---

```
MATCH p = (p1:PersonPositive:true)-[*]-(p2:Person)
WHERE ALL(r IN relationships(p) WHERE (p1.Last_Confirm >= r.Date and
p1.Last_Confirm <= r.Date + duration(days: 15)))
MATCH (p2:Person)-[:HAS]->(m:Medical_RecordRisky_Subject:true)
WHERE NOT (m)-[:REGISTERED]->(:Covid_Vaccines)
RETURN DISTINCT p1.First_Name + " " + p1.Last_Name as Positive,
p2.First_Name + " " + p2.Last_Name as Contacted, p1.Last_Confirm, p
```



Query n. 7

Given two people, one positive and one negative, find the shortest path for which they could have been in contact The negative person has only vaccine and he hasn't done a test in the last 2 days from the positive confirmation.

```
MATCH (p1:PersonPositive: true), (p2:PersonPositive:false),
p = shortestPath((p1)-[*]-(p2))
WHERE ALL(r IN relationships(p) WHERE (p1.Last_Confirm >= r.Date and
p1.Last_Confirm <= r.Date + duration(days: 15)))
MATCH (p2)-[:HAS]->(m:Medical_Record), (m)-[:REGISTERED]->(t:Covid_Tests),
(m)-[r:REGISTERED]->(v:Covid_Vaccines)
WITH p1, p2, max(t.Date) as Max_Date, count(v) as nv
WHERE (p1.Last_Confirm >= Max_Date and
p1.Last_Confirm > Max_Date + duration(days: 2)) and
nv=1
RETURN DISTINCT p1.First_Name + " " + p1.Last_Name as Positive,
p2.First_Name + " " + p2.Last_Name as Negative, p1.Last_Confirm,
Max_Date as Latest_Test
```

"Positive"	"Negative"	"p1.Last_Confirm"	"Latest_Test"
"Arnold Concepcion"	"Jeffery Sons"	"2021-10-22"	"2021-09-23"
"Dawn Irish"	"Victor Neel"	"2021-11-11"	"2020-10-17"
"Jason Kirkland"	"Beverly George"	"2021-10-07"	"2021-07-12"
"Arthur Winkler"	"Jeffery Sons"	"2021-10-21"	"2021-09-23"
"Jane Stahr"	"Roger Miller"	"2021-10-16"	"2021-07-19"
"Richard Gray"	"Janet Perry"	"2021-10-26"	"2021-02-18"
"Nadine Cooley"	"Emily Faucher"	"2021-10-23"	"2020-03-20"
"Frances Robinson"	"Beverly George"	"2021-10-05"	"2021-07-12"
"Travis Ruis"	"Cindy Schroder"	"2021-10-13"	"2020-07-13"
"Alice Stone"	"Lorraine Carter"	"2021-10-16"	"2020-06-29"

3.1.1 Query n. 8

Find all the connection a positive person had, returning nodes for graphical representation.

```
MATCH(p:Person)
WITH p, p.Last_Confirm as P_Date, duration(days:50) as TimeElapsed
MATCH relation_1 = (p)-[v1:VISITS]->(pl:Place)<-[v3:VISITS]-(:Person)
WHERE (P_Date > v1.Date and P_Date < v1.Date + TimeElapsed) and
(P_Date > v3.Date and P_Date < v3.Date + TimeElapsed) and
(v1.Date >  v3.Date and v1.Date  < v3.Date + TimeElapsed)
WITH p, P_Date, relation_1, TimeElapsed
MATCH relation_2 = (p)-[v2:VISITS]->(r:Rooms)<-[v4:VISITS]-(:Person)
WHERE(P_Date > v2.Date and P_Date < v2.Date + TimeElapsed) and
(P_Date > v4.Date and P_Date < v4.Date + TimeElapsed) and
(v2.Date >  v4.Date and v2.Date  < v4.Date + TimeElapsed)
WITH p, P_Date, relation_1, relation_2, TimeElapsed
```

```
MATCH relation_3 = (p)-[c:CONTACTS]->(:Person)
WHERE (P_Date > c.Date and P_Date < c.Date + TimeElapsed)
WITH p, P_Date, relation_1, relation_2, relation_3, TimeElapsed
MATCH relation_4 = (p)-[:LIVES]->(:Person)
RETURN relation_1, relation_2, relation_3, relation_4
```

3.2 Commands

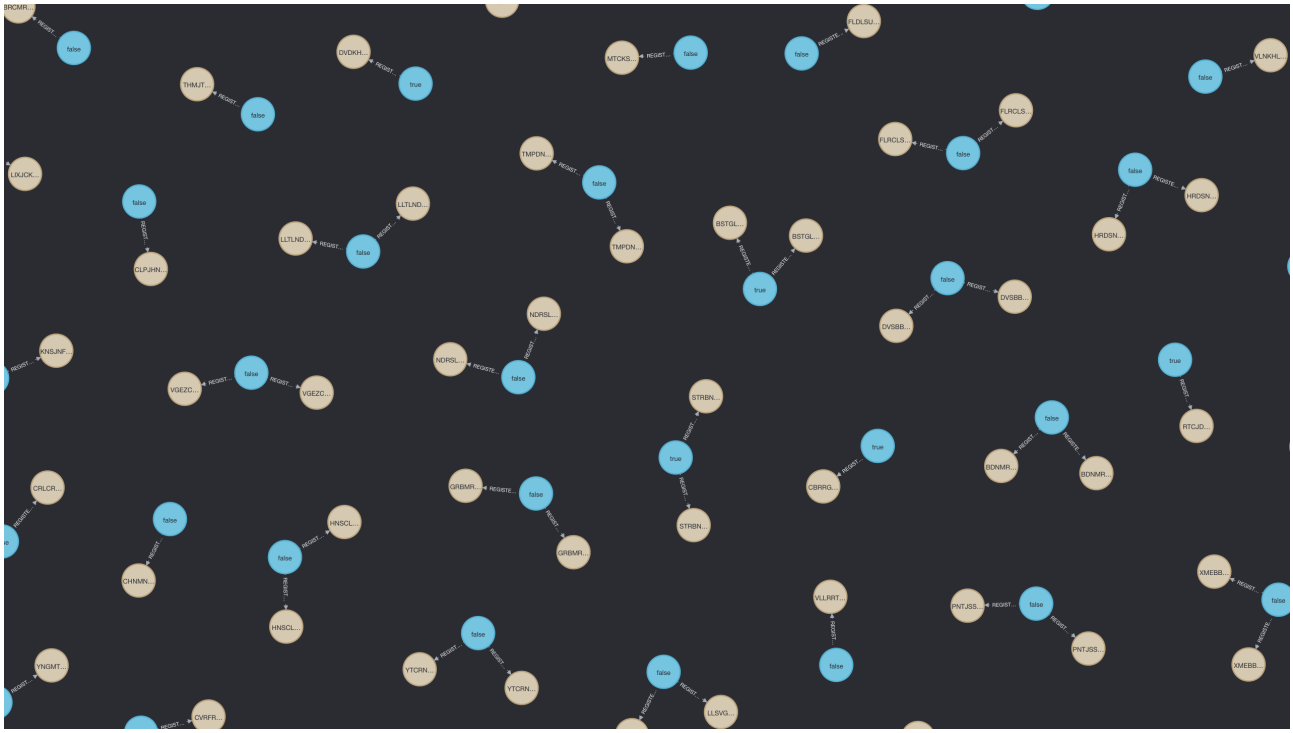
Below some commands that manipulates the database.
(NOTE) The nodes generated with "generate_dataset.py" are coerent with the assumptions, in order to get some corrupted CSV files, use "generate_inconsistent_dataset.py"

3.2.1 Command n. 1

If a person has done at least 1 vaccine, set his Vaccinated attribute to true

---

```
MATCH (p:Person)-[:HAS]->(m:Medical_Record), (m)-[:REGISTERED]->(v:Covid_Vaccines)
SET m.Covid_Vaccinated=True
RETURN DISTINCT m, v
```

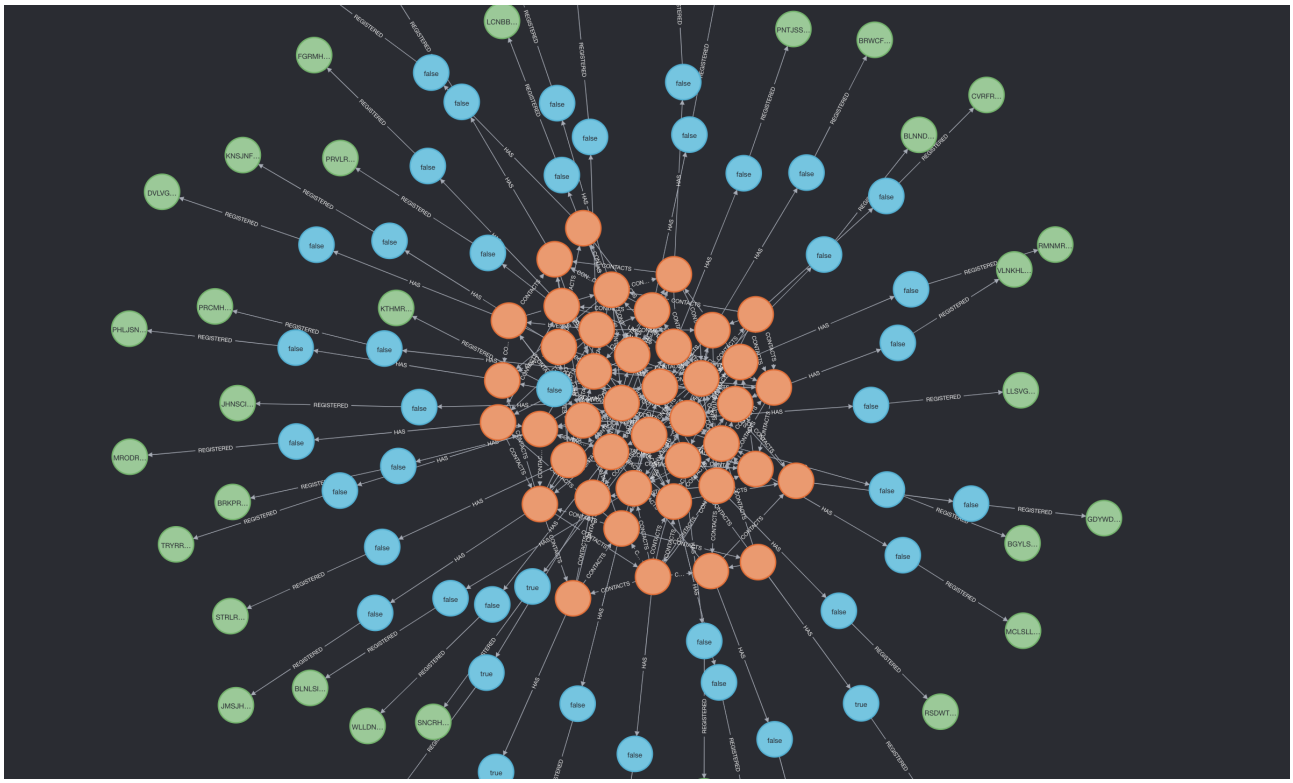


3.2.2 Command n. 2

If a person results negative in his latest test then set is Positive attribute to false.

```
MATCH (p:Person)-[:HAS]->(m:Medical_Record), (m)-[:REGISTERED]->(t:Covid_Tests)
WITH p, max(t.Date) as Max_Date
MATCH(p)-[:HAS]->(m:Medical_Record), (m)-[:REGISTERED]->(t:Covid_Tests)
WHERE t.Date = Max_Date and t.Result=False
```

```
SET p.Positive = False
RETURN p, m, t
```



3.2.3 Command n. 3

Delete all rooms never visited from any person

```
MATCH (r:Rooms)
WHERE NOT (:Person)-[:VISITS]-(r)
DETACH DELETE r
```

```
Deleted 915 nodes, deleted 915 relationships, completed after 24 ms.
```

3.2.4 Command n. 4

Delete all people without medical record

```
MATCH (p:Person)
WHERE NOT (p)-[:HAS]->(:Medical_Record)
DETACH DELETE p
```

```
Deleted 55 nodes, deleted 1537 relationships, completed after 21 ms.
```

4 User Interface Implementation

The user interface was centered around 3 pillars:

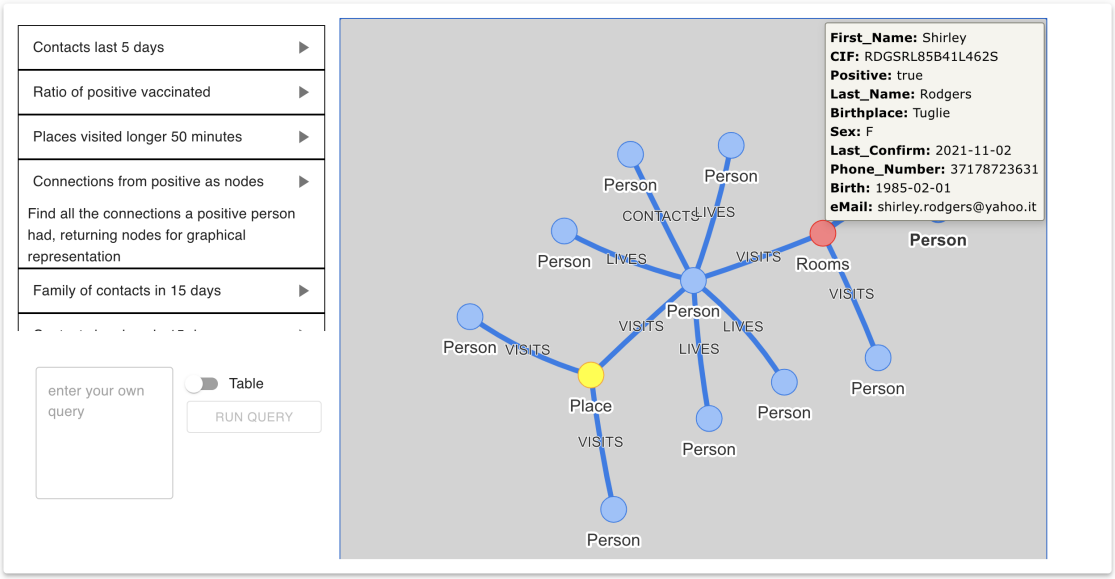
1. **Simplicity.** We set out to build a simple, streamlined web application, in order to gain familiarity with the tools that interact with Neo4J. That said, we wanted to avoid increasing the complexity of the project, so we strove to keep it as straight forward as possible.

2. **Usefulness.** Although the User Interface is simple, it is not useless. In this context, we decided our measure of utility would not be a "commercial" one, like the value one could generate from our product, but the knowledge we would gain by building it.

3. **UX.** Because of the simplicity of the webapp, we thought the least we could do was to make it easy to use. We hope we succeeded!

The UI is divided into 3 sections: in the top left, the query list is available. Each one has a description hidden under a collapsable pane, a simple button to run it and a name. In the bottom left, we included a way to run personalized queries, leaving it to the user if they wanted to see the result as a graph, or as a table. Lastly, on the right half of the screen, either a graph or a table collecting results is shown.

EXPLORER



The tools we leveraged to code it are:

1. **React.** We believe it needs no introduction; one of our project members had already worked with it, so we decided to leverage his experience to fast-forward development.
2. **Material UI.** We needed a simple, preconfigured library for common UI elements, with React components included out of the box.
3. **Use-Neo4J.** Luckily, we weren't the first one to bind a React application to a Neo4J database (considering the popularity of the GRANDstack, this wasn't news to us). We used this simple library to manage drivers and simplify our code.
4. **NeoVis.js** This simple repo wraps a Neo4J driver together with a vis.js visualization method; because of version clashing, we were forced to use an older version, which didn't allow us to use an external driver; this means some of our requests are doubled.

5 Conclusion and possible improvements

In this report, we outlined design and implementative choices behind the first part of the SAMBUD Fall 2021 project at Polimi. The project has been very interesting both from the theoretical part and the practical part. From the practical point of view it has permitted to understand better and elaborate on Neo4j and Cypher. From the theoretical point of view it has been an interesting challenge to create an ER model and then a graph database that could be used to reason on a real problem. It was interesting working on the differences between a relational database and a graph database mostly thanks to the features of a graph. The queries were created to be easy to understand but also trying to use and highlight the potential of Neo4j. The graphic interface could be improved, but it fits our initial goals.