



# Project report - Neo4j

## Introduction

First delivery of the project work for systems and methods for big and unstructured data.

Members of the group 5:

- Alice Brugnoli, 10608045.
- Ginevra Iorio, 10607321.
- Francesco Gonzales, 10614882.
- Leonardo Caponi, 10618575.
- Andrea Ceresetti, 10633584.

## Problem Specification

The task was to build an information system for managing pandemic data for a certain country. In this first delivery we had to keep trace of the contacts between people, to monitor the viral diffusion, designing and storing a query graph data structure in a NoSQL DB.

In our database we had to record:

- People and their connections through contact tracing apps and explicit data collection collected in public places like restaurants, theatres, hospitals and so on.
- Time and place of the contact, when it is possible to be determined univocally.
- Personal data of each person including vaccines, covid tests and contagion date and place when it is possible to be determined univocally.

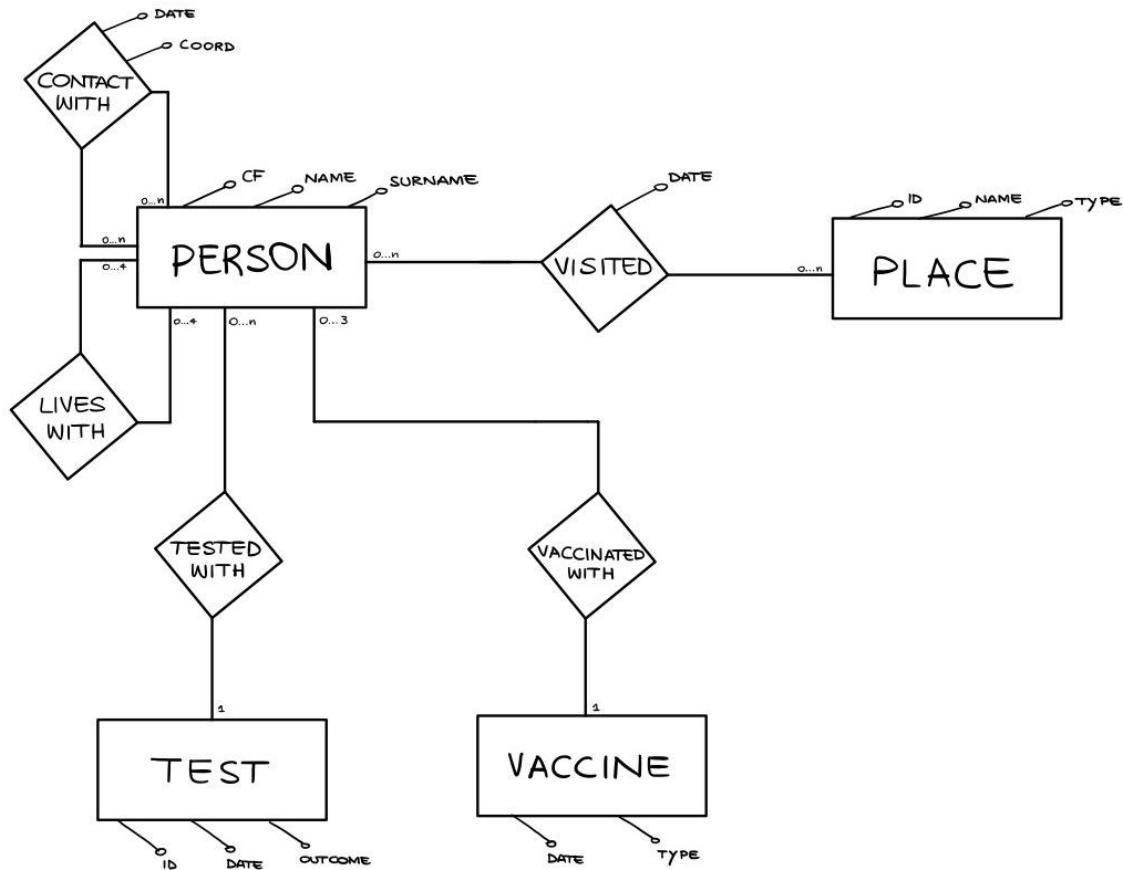
## Hypothesis

- Persons are identified by the CF.
- Public places collect people's data and time of entry.
- Contagion occurs if a person becomes covid positive after being in contact with a positive person within a selected time frame (14 days).
- Contagion's date and place (or a list of potential contagions) can only be told when a person was in contact with one (or more) positive persons within a selected time span.
- If someone results positive to a test, after 14 days he becomes negative.

## ER diagram

- PERSON: Persons visit places, get in contact or live with other people, do covid tests and get vaccinated.

- PLACE: Places register people's entries.
- TEST: Tests are done to discover positivity, that lasts 14 days.
- VACCINE: Vaccines are done of a certain type and they are maximum three per person.



## Dataset description

We created the dataset using two different methodologies for nodes and for relationships.

We used a free online service [1] to create the csv files of our nodes, namely people, places, tests and vaccines. We chose this service to create our entities since the data is realistic (person names, cf, emails, places names etc) and it's fast and easy to use.

To create relationships we wrote some scripts to generate the relationships between the previously created nodes, retrieved by importing the csv and writing the commands to a file. The full code is available on github [2].

# Queries & Commands

## Queries:

1. Given a positive, find all the places he has visited in the previous 14 days.

```
MATCH (t:Test{outcome:"true"})<-[:TESTED_WITH]-(p:Person)-[v:VISITED]->(q:Place)
WHERE date(t.date) >= date(v.datetime) AND date(t.date) <= date(v.datetime) +
duration('P14D')
RETURN q.name as Place, p.cf as Positive
```

"Place"	"Positive"
"Runte-Barton"	"SZYQDZ09I45Z436S"
"Bauch Group"	"TQAFXT86B64C295V"
"Steuber, Kris and D'Amore"	"EZFUPR51I51F317P"
"Bradtke-Stroman"	"FSNSCX98D08U537M"
"Leannon Group"	"KREVSW96M81L916C"
"Homenick, Hodkiewicz and Feil"	"KREVSW96M81L916C"
"Barton, Botsford and Bartoletti"	"JQONKQ35Y07T084M"
"Okuneva Group"	"PRZOUA03O78N999U"
"Dicki, Heidenreich and Hand"	"KGPZZR92J80G087A"
"Greenholt, Greenholt and White"	"JIGMPB31X89O800K"
"Leffler, Bashirian and Crooks"	"PRWSIN48H89V002Q"
"O'Connell, Zulauf and Cummings"	"LRDZSK01C50Z472Z"
"Smitham, Zulauf and Kertzmann"	"BVQLAP07C43F810F"
"Nienow and Sons"	"NSCROS26O78A472L"
"Pfeffer, Little and Flatley"	"DHOBXM29Q45U561N"
"Hessel Group"	"QFYVOX68G85G213Y"
"Rempel LLC"	"NTOKCS30O51Q114N"
"Fritsch LLC"	"TNBSNT17Y02Z226T"

"Smitham, Satterfield and Schowalter"	"HZPLES08D70A107L"
"O'Kon-Williamson"	"LSPISN75J61C569C"
"Heathcote, Prohaska and Hintz"	"FDKCFE38Z46A574F"
"Klocko and Sons"	"ZUSOPJ25X13R176R"
"Wilkinson, Rice and West"	"ZESJMF18Z67Z397S"
"Kuhlman-Maggio"	"MUJFVZ14A26F980C"
"D'Amore-Parker"	"ETIWJC53M18X466W"
"Larkin-Rosenbaum"	"WNAOTT83Y66Q493Q"
"Bruen, Walsh and Kunde"	"LIQPYE74B39A817B"
"Hansen-Pacocha"	"VYKPUP46H99O998T"
"Leffler-Metz"	"PFNDRU26B48L685S"
"Cruickshank, Considine and Konopelski"	"PFNDRU26B48L685S"
"Runte Inc"	"WWYLXR77A38X851T"
"Greenholt, Greenholt and White"	"DMYYWR93C88X054P"
"Kunze Group"	"PRIJTS84S37M686K"
"Bradtke-Stroman"	"PRIJTS84S37M686K"
"Hagenes, Veum and Weber"	"YDOXPB03H90L763J"
"DuBuque Group"	"XAHONJ48B44S919N"

- Find out how many families have had an infected member who, however, has not infected anyone else within the family.

```

MATCH (p)-[:LIVES_WITH]-(p1:Person)-[:TESTED_WITH]->(t1:Test{outcome:"true"}),
(t2:Test{outcome:"true"}), (t3:Test{outcome:"true"})
WHERE date(t1.date) <= date(t2.date) AND date(t1.date)+duration('P14D') >= date(t2.date)
AND date(t3.date) <= date(t1.date) AND date(t3.date)+duration('P14D') >= date(t1.date)
AND NOT (p)-[:TESTED_WITH]->(t2)
AND NOT (p)-[:TESTED_WITH]->(t3)
RETURN count(DISTINCT p1)

```

"count (DISTINCT p1) "
66

- Find vaccinated people who have been in contact with a positive one and who have not been infected (negative test or no test).

#### MATCH

(v:Vaccine)<-[:VACCINATED\_WITH]-(p1:Person)-[c:CONTACT\_WITH]->(p2:Person)-[:TESTED\_WITH]->(t2:Test{outcome:"true"}), (t1:Test{outcome:"true"})

WHERE date(t1.date) >= date(t2.date)

AND date(t2.date) >= date(c.date)

AND date(t1.date) <= date(c.date)+duration('P14D')

AND date(t2.date) <= date(c.date)+duration('P14D')

AND NOT (t1)<-[:TESTED\_WITH]-(p1)

RETURN DISTINCT p1.cf as Vaccinated, p2.cf as Positive

"Vaccinated"	"Positive"
"EFMTMI35V52G513E"	"IJBZK30N94J122F"
"KGPZZR92J80G087A"	"GPIVMX64W67E888E"
"CSZMHE60C15E619J"	"EZFUPR51I51F317P"
"PWEJSH54E58R793A"	"SZILTQ00Y74K544S"
"MXRNNI42R03D357N"	"UCNXYO90R54X973R"
"RKNPOK93U95N569Z"	"YLIUMM06Y75S403K"
"BVQLAP07C43F810F"	"WNAOTT83Y66Q493Q"
"RKLSYZ80R80O264I"	"QTBEBV18G11D929W"
"CSDYSC69X99H638F"	"QTBEBV18G11D929W"
"BVQLAP07C43F810F"	"PRZOUA03O78N999U"
"VVYLGR56J79N532C"	"LCOXYA13U67Y940F"
"FUATVU11A86B274R"	"JIGMPB31X89O800K"
"OTAQXN42Z20G008H"	"FGTBLP95Y25Z937A"

"IVWLL022V33V253G"	"FGTBLP95Y25Z937A"
"MUWXPZ69C70Z196X"	"IJTFNQ91F07B168Q"
"VZNPAU25G55O498I"	"RUJPKM92G36B605W"
"JMBNQT75E24Q340M"	"KXVCNY21E03E342F"
"MBEPML91W95O926M"	"USMTTP75V05C049F"
"AHDNOG18Z61P146S"	"IDZYJC67T91M874I"
"QKCNXI23C49R374B"	"JZBGEJ33T05F972Q"
"CYMAFE30J89S493T"	"DQVGLS68N03O979N"
"CGFZNL33S86K047P"	"IWEPM31M67O918L"
"FKAQHI96X96C689M"	"JHASQD34W44O958J"
"VKBTNL74B77Y979D"	"TRJDFI52X07C879M"
"VCIQQY92G94O821T"	"JXSCRX60G68K188P"
"WMEJLB15E25O127Z"	"CGFZNL33S86K047P"
"RKLSYZ80R80O264I"	"YCBJUZ64R07D577K"
"DILCZM72Z72V479F"	"IIFGLS15Z82W774S"
"HDLKOW31F68Z927Z"	"ZFAPFZ83O86L216N"
"XAXOWQ64A23Q502J"	"PFIEBD19G59H420Z"
"SWWXXR79R27V292F"	"FAQYPB71H66M447Y"
"QBBNAD41S86G075S"	"FAQYPB71H66M447Y"

4. Find the people who were at the bar on August 15 and found themselves positive over the next 14 days.

**MATCH**

(t:Test{outcome:"true"})<-[:TESTED\_WITH]-(p:Person)-[v:VISITED{datetime:datetime("2021-08-15")}]>-(q:Place{type:"Bar"})

**WHERE** date(t.date) <= date(v.datetime)+duration('P14D') **AND** date(t.date) >= date(v.datetime)

**RETURN** p.cf

"p.cf"
"YDOXPB03H90L763J"

- Find all the positive people of whom we are sure of the identity of his infector (i.e. he has been in contact with only one positive).

```

MATCH (t1:Test{outcome:"true"})<-[:TESTED_WITH]-(p1:Person)
-[r:CONTACT_WITH]->(p2:Person)-[:TESTED_WITH]->(t:Test{outcome:"true"})
WITH count(p2) AS c, p1 AS a, r AS contact, t AS test, p2 AS b, t1 AS test1
WHERE c = 1
AND date(contact.date) >= date(test.date)-duration('P14D')
AND date(contact.date) <= date(test.date)
AND date(contact.date) <= date(test1.date)
AND date(contact.date)+duration('P14D') >= date(test1.date)
RETURN a.cf as Positive, b.cf as Infector

```

"Positive"	"Infector"
"VZNPAU25G55O498I"	"RUJPKM92G36B605W"
"WMEJLB15E25O127Z"	"CGFZNL33S86K047P"
"YYDRHH35P66H732L"	"EZFUPR51I51F317P"
"USMTTP75V05C049F"	"PRWSIN48H89V002Q"
"QBBNAD41S86G075S"	"FAQYPB71H66M447Y"

## Commands:

- Create a new test.

```

MATCH (a:Person{cf:'URVYVT81K23T745R'}), (b:Test{id:'1462'})
CREATE (a)-[:TESTED_WITH]->(b);

```

Created 1 relationship, completed after 25 ms.

- Update the name of a bar.

```

MATCH (p:Place{id:'19'})

```

```
SET p.name = 'Novansky'  
RETURN p
```

```
{  
  "identity": 1018,  
  "labels": [  
    "Place"  
  ],  
  "properties": {  
    "name": "Novansky",  
    "id": "19",  
    "type": "Bar"  
  }  
}
```

3. Delete negative tests older than 10 months.

```
MATCH (t:Test{outcome:"false"})  
WHERE date(t.date) <= date()-duration('P10M')  
DETACH DELETE t
```

Deleted 388 nodes, deleted 388 relationships, completed after 65 ms.

## UI description

We created the UI using Neovis.js [3] tool, which we downloaded from the Neo4j Open Source Contributions Ecosystem. This tool provided us with all the functions required to create an html webpage connected to our database, showing its graphic on screen. Using JavaScript and HTML programming languages, we then implemented all the UI's functionalities:

- A box that allows to make any Cypher query to the database
- Buttons that show the graphical results of our 5 queries
- The possibility to add, delete and update a node or relationship

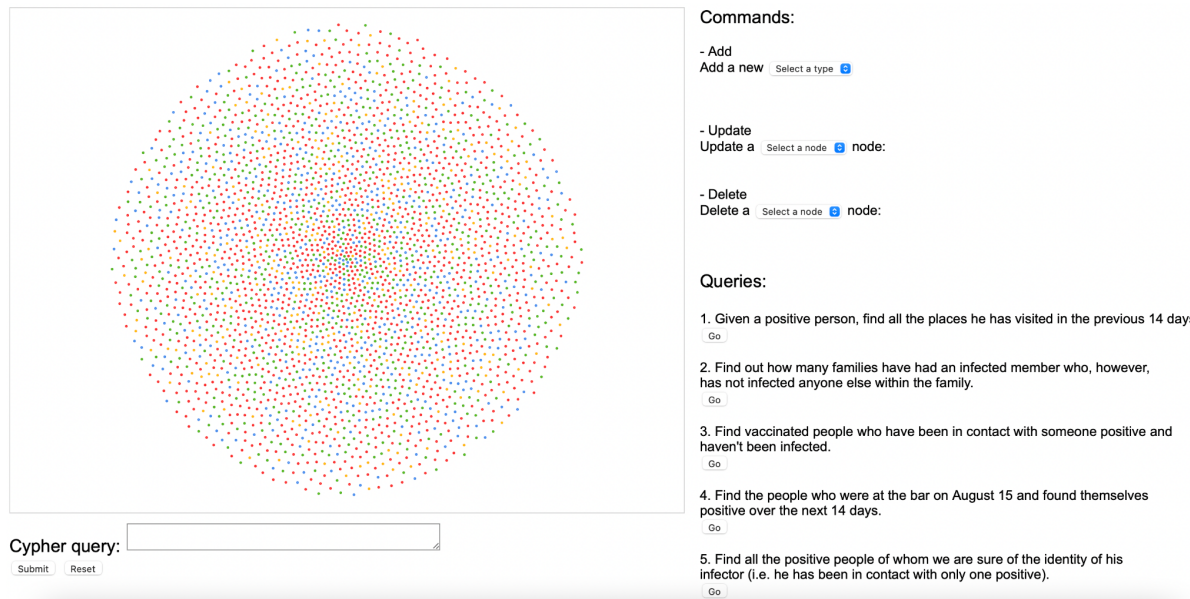
## User Guide

To load our dump into the DB you can run the following command

```
bin/neo4j-admin load  
--from=path/to/directory/neo4j-mongo-hadoop/neo4j/neo4j-SMBUD.dump
```



After loading the dump and running neo4j on port 7687, you can simply visit



**Commands:**

- Add  
Add a new  node:
- Update  
Update a  node:
- Delete  
Delete a  node:

**Queries:**

1. Given a positive person, find all the places he has visited in the previous 14 days.
2. Find out how many families have had an infected member who, however, has not infected anyone else within the family.
3. Find vaccinated people who have been in contact with someone positive and haven't been infected.
4. Find the people who were at the bar on August 15 and found themselves positive over the next 14 days.
5. Find all the positive people of whom we are sure of the identity of his infector (i.e. he has been in contact with only one positive).

Cypher query:

<https://fulcus.github.io/neo4j-mongo-hadoop/neo4j/ui/index>

In the user interface there are mainly 4 possible commands for the user:

1. **Cypher query:** allows to insert a query in a completely free way.

Cypher query:

2. **Commands:** the user is guided in creating, updating or deleting a node or a relation. Based on the selected action, the application requests the necessary data, generates the query and executes it.

**Commands:**

- Add a new  node:

Create a new  node:

firstName:  lastName:

cf:  email:

- Update/Delete

Find a  node:

date:  outcome:

3. **Queries:** in this case the user can choose among the 5 queries proposed in the previous section, testing their operation.

### Queries:

1. Given a positive person, find all the places he has visited in the previous 14 days.

2. Find out how many families have had an infected member who, however, has not infected anyone else within the family.

3. Find vaccinated people who have been in contact with someone positive and haven't been infected.

4. Find the people who were at the bar on September 22 and found themselves positive over the next 14 days.

5. Find all the positive people of whom we are sure of the identity of his infector (i.e. he has been in contact with only one positive).

4. **Reset:** resets the view showing in the pane all the nodes of the db

## References and sources

[1] Mockaroo <https://www.mockaroo.com/>

[2] Github repo <https://github.com/fulcus/neo4j-mongo-hadoop>

[3] Neovis.js <https://github.com/neo4j-contrib/neovis.js/>