Group Project Part 2 - MongoDB

of Systems and Methods for Big and Unstructured Data Course (SMBUD)

held by

Brambilla Marco

Tocchetti Andrea

Group 14

Banfi Federico 10581441

Carotenuto Alessandro 10803080

Donati Riccardo 10669618

Mornatta Davide 10657647

Zancani Lea 10608972

Academic year 2021/2022

Contents

1 Problem Specification

During the sanitary emergency due to the Covid-19 pandemic, many programs and applications developed thanks to the use of Big Data proved to be particularly effective in different settings and scenarios, such as the hospital administration, the hospitalizations organization or the analysis of data relating to various clinical cases. Among the various areas that have been supported by these technologies there is also that which concerns the tracking of populations belonging to a given geographical area and the collection of all information regarding the tests carried out and the vaccination status.

Our project aims to create an information system suitable for this specific use case and to do this it is necessary to design a database that allows us to store large amounts of data derived from heterogeneous sources and to carry out targeted queries useful for different purposes. The NOSQL document-based approach, known for being based on managing data saved in JSON-like documents, is the optimal one in this case and MongoDB is the open source database management software that is best suited to accomplish this task thanks to the considerable scalability guaranteed by the automatic data sharding and ease of use thanks to the dynamic schemes developed starting from the archived documents.

2 Hypotheses

During the design phase,some considerations were made regarding how to structure and implement the database to obtain a solution that was effective and performing but at the same time consistent with the real current scenarios. First of all, two types of documents have been created: Certification and AuthorizedBody; the first represents a sort of "medical chart" containing the following fundamental information for each individual:

1. list of tests she/he has undergone,
2. list of vaccine doses that have been administered,
3. personal details (such as name, surname, date of birth, etc.),
4. one or more emergency contacts to call in case of need.

Within the documents stored in the database for each of these four fields, subfields have been provided (as shown in the .JSON schema) to manage the various data with MongoDB in order to execute specific queries and commands.

The second document contains details related to the Authorized Bodies, i.e. institutional places where it is possible to get vaccinated and/or where

one can undergo a test, the fields are based on specific assumptions about where these places are located and the healthcare personnel working there.

At last, in order to record a greater number of elements to be processed within the dataset, while ensuring the meaningfulness and consistency of the information stored, some assumptions and limitations have been established in the creation of the data managed by the database, therefore:

- each Authorized Body is associated with a document identified by a unique ID, which is directly referred to in the test/vaccine lists of the Certification document;
- each test/vaccine was associated with only one member of the healthcare personnel who represents a sort of "responsible" for that given administration;
- for the sake of simplicity in data management, it was decided that for each Authorized Body there should be a list consisting of at most 5 members of the healthcare personnel, "responsible" for the various tests/vaccines to which they are associated;
- each person can come from any Italian location, but only Authorized Bodies belonging to the city of Milan with the relative coordinates have been examined for the purpose of performing more in-depth analysis on the data;
- vaccines were administered throughout the last year while tests during the last month.

3 ER diagram

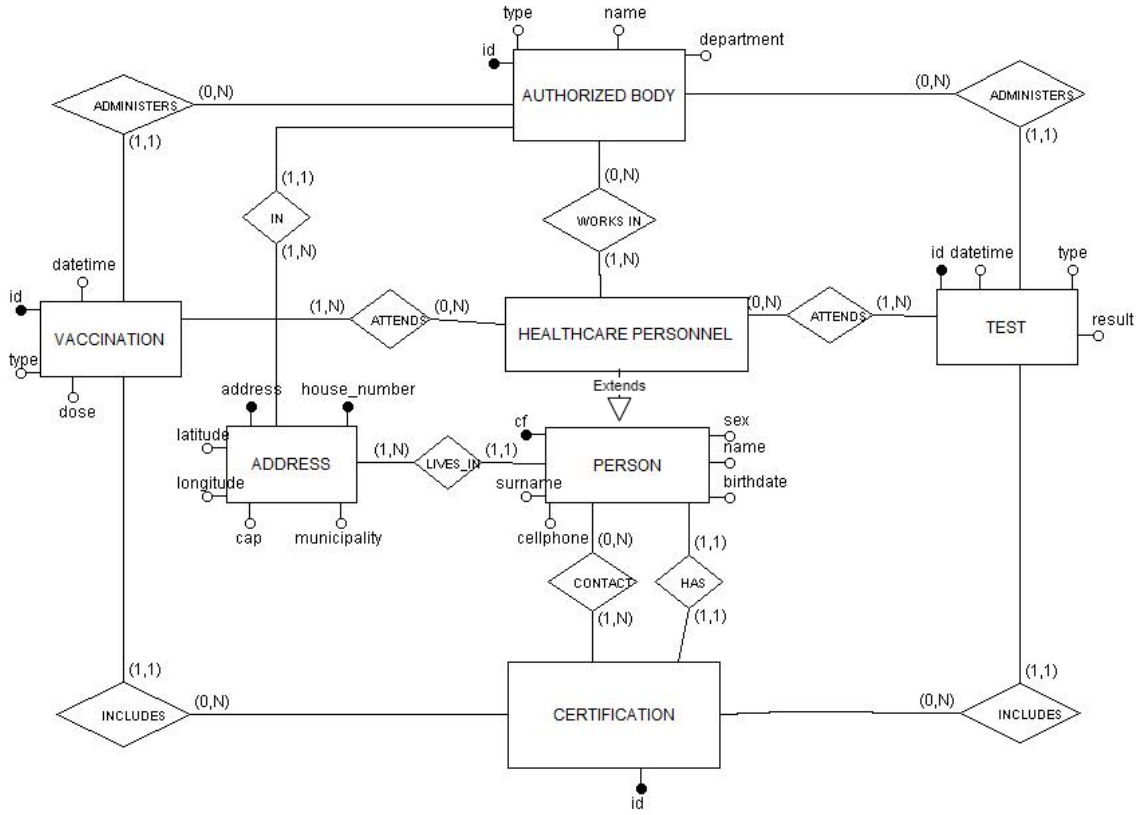

Figure 1: E-R Diagram

Starting from the considerations previously exposed regarding the implementation hypotheses, we have drawn an ER diagram (**Figure 1**) which includes 7 different entities and 4 many-to-many relationships described below in the logical model:

- **Address**(Address, HouseNumber, CAP, Latitude, Longitude, Municipality)
- **AuthorizedBody**(ID, Department, Name, Type)
- **Certification**(ID)
- **HealthcarePersonnel**(CF, Birthdate, Cellphone, Name, Sex, Surname)
- **Person**(CF, Birthdate, Cellphone, Name, Sex, Surname)
- **Test**(ID, Datetime, Result, Type)
- **Vaccination**(ID, Datetime, Dose, Type)
- **AttendsTest**(HealthcarePersonnel.CF, Test.ID)
- **AttendsVaccination**(HealthcarePersonnel.CF, Vaccition.ID)
- **Contact**(Certification.ID, Person.CF)

- **WorksIn**(AuthorizedBody.ID, HealthcarePersonnel.CF)

The **Person** entity describes each individual and their personal data, **HealthcarePersonnel** is a specialization of **Person** and includes all the people (doctors, nurses, pharmacists, etc.) who work (as indicated by the many-to-many **WorksIn** relationship) at an **AuthorizedBody**. Both the **Test** and **Vaccination** entities are linked to **HealthcarePersonnel** by the many-to-many **AttendsTest** and **AttendsVaccination** relationships, while the link with **AuthorizedBody** is given by the one-to-many **AdministersTest** and **AdministersVaccination**. Each **Person** can have one and only one (one-to-one relationship **Has**) **Certification**, that contains information regarding how many and which tests a person has undergone and whether or not he has received one or more doses of the vaccine, in fact, **Certification** is in a one-to-many relationship with both **Test** and **Vaccination**, through **IncludesTest** and **IncludesVaccination**, respectively. Furthermore, each **Certification** must also include for each **Person** a sort of "emergency contact" with its details as evidenced by the many-to-many **Contact** relationship. Finally, to save information relating to the geographical location of persons and authorized bodies, an **Address** entity was added with specific attributes, connected to **Person** and **AuthorizedBody** respectively by the one-to-many **LivesIn** and **In** relationships.

4 Dataset description

One of the most critical parts of working with Big Data is managing large amounts of data collected in large datasets. To test and simulate the use of the database for contact tracing activities, some sample datasets were generated, saved in .csv format and imported into Neo4j through the command: `LOAD CSV FROM "file:  ///file.csv" AS ....`

Each dataset is divided into various fields that trace the structure of the tables expressed in the ER model, each one was generated randomly through Python scripts (you can find these ones into the folder `"db"`) and to experiment and perform at best the possible tests on the queries and commands that can be executed thanks to Neo4j the number of entries foreseen for each dataset is in the order of magnitude of the hundreds, as a whole, starting exclusively from the data loaded from the datasets, the database provides:

Homes	101	Residence Relationship	300
People	300	Meetings	726
Public Places	20	Public Places Visits	600
Vaccine Types	4	Vaccinations	526
Test Types	2	Tests	450
TOTAL NODES	**427**	**TOTAL RELATIONSHIPS**	**2602**

5 Queries and Commands

The correct functioning of the information system involves the implementation of some essential commands and queries for the database in order to properly support the app and to ensure the right execution of searches among the data available for statistical or practical purposes.

First of all you need to load the .csv files with the query you can find following the path `"documents/importDB.txt"`

5.1 Queries

5.1.1 Find how many people went without greenpass in a public place

This query allows us to find all the people that went in a public place without the GreenPass and the datetime related.

The output is given by:

- The name and the surname of the person
- The entrance time in the public place
- The name of the public place

5.1.2 Find who lives with an infected individual

This query allows us to detect a "family contact" with an infected person and obtain the people that need to be quarantined.

The output is given by:

- The house identifier
- The name and the surname of the person
- The name and the surname of the infected cohabitant

5.1.3 Find public place contact with infect people

This query allows us to find the people who went in the same public place (e.g. restaurant, cinema,...) at the same time with infected people starting from 15 days before their positive test.

The output is given by:

- The name and the surname of the person
- The name and the surname of the infected cohabitant
- The datetime of the entrance of the health person
- The datetime of the entrance of the ill person
- The public place

5.1.4 Find who got vaccinated in a temporal range

This query allows us to find all the vaccinated people in the temporal range of October 2021.

The output is given by:

- The name and the surname of the person
- The type of the vaccine
- The datetime of the vaccine

5.1.5 Statistical analysis of the vaccination campaign

This query allows us to compute an analysis on the number of the vaccinated people (and the percentage) grouped by age ranges.

The output is given by:

- Total people
- Total vaccinated people (with %)
- Vaccinated people in age ranges (with %)

5.2 Commands

5.2.1 Federico moves in an other house (CREATE)

In this command we simulate a person moving in an another house.

5.2.2 Delete all the public place records older than 1 year (DELETE)

In this command we simulate the need of removing old records that are not useful anymore.

5.2.3 Name change of public place (UPDATE)

In this command we simulate the need, of a public place manager, of changing name of the activity.

6 UI description & User Guide

6.1 UI description

The design of the information system ended with the development of *ContagionShield*, an application connected to the database with a simple GUI that is very intuitive and easy to use. The UI was created with the Python programming language by means of the libraries: *PySimpleGUI* for the creation of the graphical interface, *Py2neo* to work with Neo4j through the syntax offered by Python and *pandas* to manage the analysis and manipulation of data, you can find these ones into the folder `"contagionshield"`.

As you can see from **Figure 3**, the main screen of the application is divided into two sections: one on the left that allows you to create customizable queries by choosing date, place, time interval and whether to display the vaccinated, non-vaccinated or with negative test, in **Figure 4** there is an example query with the results obtained; the other side section on the right (**Figure 5**) presents some predefined queries, some of them follow the ones specified and described in Chapter 5 of the report.

Finally, as shown in **Figure 6**, it is possible to execute some of the commands already presented in Chapter 5 by reaching the appropriate section of the application by means of the `"Commands"` button in the lower left corner of the main screen. The executable commands allow you to create new meetings between several people by indicating their social security numbers, the date and time, create new visits to public places by specifying who went to a given place and when and in the end you can also delete all the visits to public places recorded in the last year.

6.2 User Guide

In order to run *ContagionShield* it is necessary to verify some requirements and to perform some actions:

- At first you have to check if you have the right Python version installed by using the command: `python --version`, if not you could download it from the official website: https://www.python.org/
- Then make sure your local database is at `localhost:7687` and that it has `"smbud"` as password
- Install the required packages (in the "contagionshield" folder): `pip install -r requirements.txt`
- Finally make it run by navigating to the folder where you have been saved the materials, then into `"contagionshield"` folder and run: `python contagionshield.py`
- From there you can execute queries about the collected data

7 References & Sources

[1] Course Slides

[2] https://pysimplegui.readthedocs.io/en/latest/call

[3] https://py2neo.org/

[4] https://neo4j.com/docs/cypher-manual/current/

[5] https://neo4j.com/developer/python/

[6] http://iniball.altervista.org/Software/ProgER

[7] https://neo4j.com/developer/cypher/

[8] https://pandas.pydata.org/docs/