Databases
Informatics and Computing Engineering Department
Faculty of Engineering, University of Porto

Vaccination Monitoring Programme

A Research and Application on a Minimal Vaccination Campaign Database

Luís Tavares
Márcio Duarte
Nuno Costa

email: {up201809679, up201909936, up201906272}@fe.up.pt

2nd class, Group 205

May 23, 2021

Contents

1. Context

Nowadays, vaccination presents itself as one of the most critical problems in modern society. Many countries, over the years, have developed a National Vaccination Programme to prevent epidemics and improve citizens' health care. Portugal has had one since 1965, where a universal and free programme was born. Programmes like these require large amounts of data and adequate data structures to store reliable information. This project aims to describe a minimal vaccination campaign database.

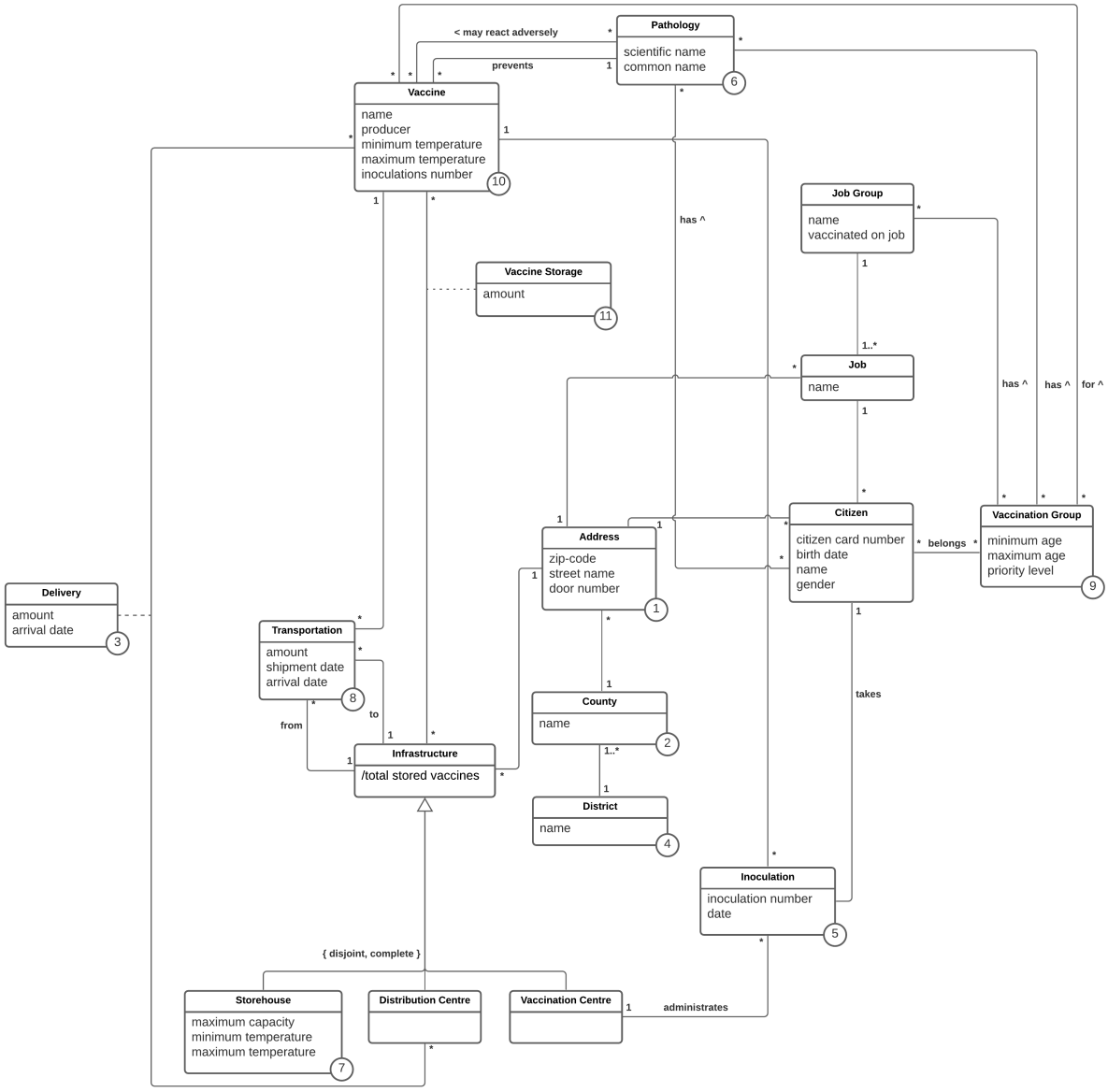
In this vaccination campaign database, citizens and vaccines are central entities. Citizens are identified by their citizen card number and hold sociodemographic data. On the other hand, vaccines provide information about the pathology it prevents and the ones it might react adversely to. Distributions centres store produced vaccines. Later, transportations between distribution centres and storehouses or vaccination centres may occur. These transportations only carry a single type of vaccine due to conservation restrictions (*e.g.* storage temperature range). After arriving at a storehouse, vaccines remain preserved in their storage conditions. Storehouses act as a middleware between a distribution centre and a vaccination centre. However, transportations may occur directly between a distribution centre and a vaccination centre. As some shipments have a long route, vaccine packages might travel between multiple storehouses to reach the final destination, allowing transportations to arise between two storehouses.

Vaccinating a population is particularly challenging. To ease the vaccination process, citizens have a defined vaccination group. A group for a vaccine is responsible for delineating its allocation criteria and its priority. After having a correctly structured collection of vaccination groups, the vaccination process starts. During the vaccination process, groups might suffer changes, allowing large groups to split into smaller ones.

The vaccination of a citizen takes place at a vaccination centre. The vaccination centre assigned to a citizen is dependent on sociodemographic data. For general purposes, the vaccination centre of a citizen is the closest to their address. However, the allocation criteria change according to the citizen vaccination group. An inoculation registry is maintained, holding information about the citizen, the vaccine and the vaccination centre.

2. Conceptual Modeling

2.1 UML Diagram



2.2 Class Definition and Restrictions

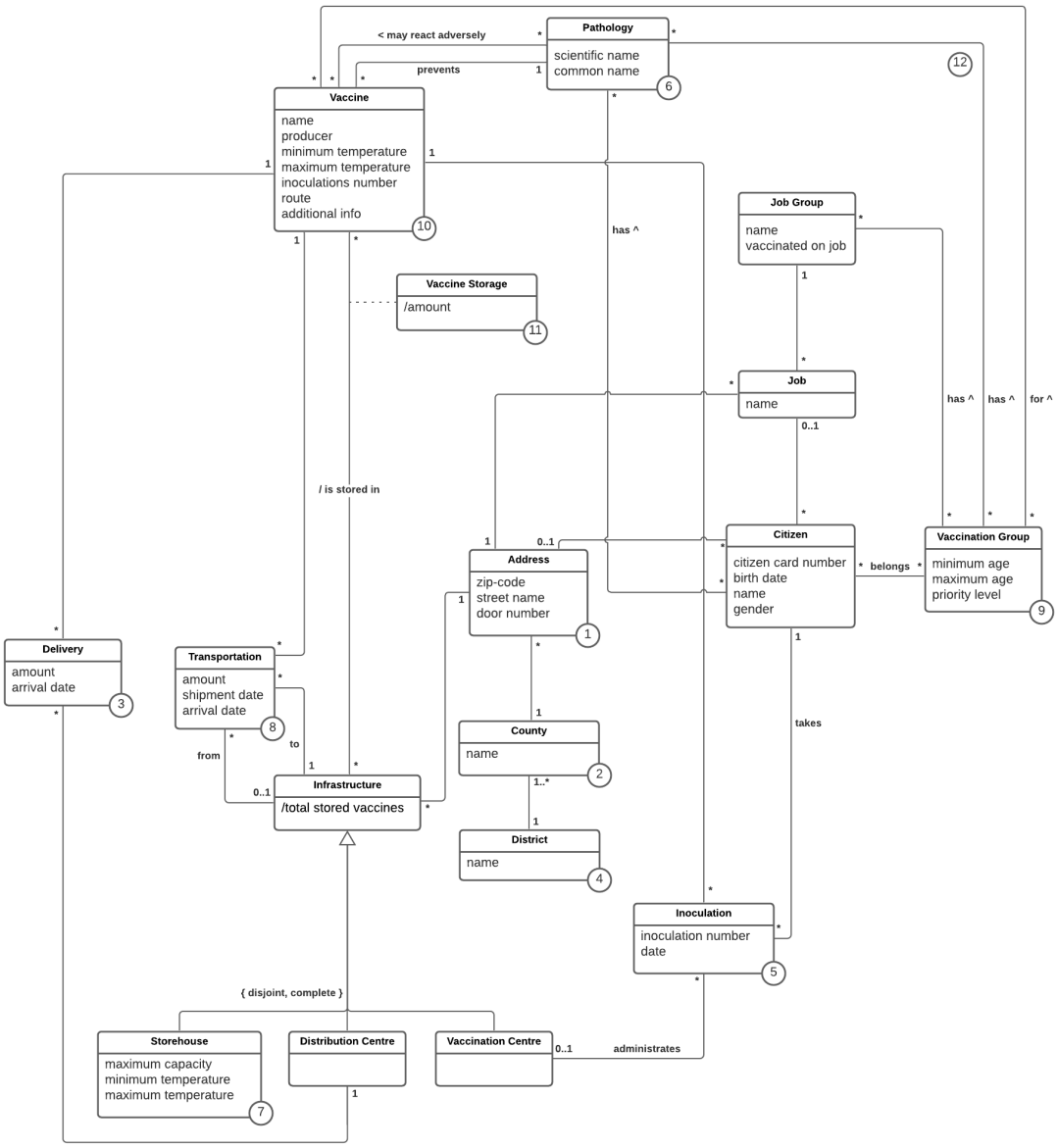
Name	Definition	Restrictions
Vaccine	Defines a vaccine by its name, producer, the number of inoculations it requires and the temperature range at which it must be reserved. The pathology it prevents, as well as the pathologies it may react adversely to, are also defined.	(10) a) inoculations num > 0 b) maximum temperature $\geq$ minimum temperature
Infrastructure	Defines an infrastructure by its address and the total (calculated) amount of vaccines it stores. It also keeps track of all of its vaccines' stock. It may receive or deliver transportations of vaccines.	None
Storehouse	It is a generalization of an Infrastructure. Represents a storehouse and, as such, has a given max capacity. It also defines at what temperature range vaccines can be stored.	(7) a) maximum temperature $\geq$ minimum temperature
Distribution Centre	It is a generalization of an Infrastructure. Represents a main centre of distribution and, as such, holds the first stop for all vaccines.	None
Vaccination Centre	It is a generalization of an Infrastructure. Defines the place where inoculations are taken.	None
Vaccine Storage	Defines the stock of a given vaccine in a given infrastructure.	(11) a) amount ≥ 0
Transportation	Defines the amount of vaccines that are being transported, as well as the shipment and arrival date, the vaccine being transported and the infrastructure from which they are being taken and to which they are to be delivered.	(8) a) amount > 0 b) Transportation can't be held to Distribution Centre

Delivery	Defines a shipment to a distribution centre of a certain vaccine. It holds the amount delivered and the date of arrival.	(3) a) amount > 0
Inoculation	Defines a given inoculation of a certain vaccine of a citizen in a certain vaccination centre. It is also described by its number (first, second (...) take of a given vaccine) and the date.	(4) a) $1 \leq$ inoculation number $\leq$ inoculations num of the associated Vaccine
Pathology	Defines a pathology, which is defined by its scientific name and common name. It may be prevented by or it may react adversely to a vaccine.	(6) a) scientific name must be unique
Vaccination Group	Defines a group by the minimum and maximum ages, as well as its priority level, represented by a number. The lower that number is, the higher the priority of that group. It is formed for a given vaccine and contains citizens with a certain job group and a certain set of pathologies.	(9) a) $0 \leq$ minimum age $\leq$ maximum age b) priority ≥ 0
Job Group	Defines a job group. A job group contains jobs of the same area (medical job group $\rightarrow$ nurse, doctor..). It is defined by its name and whether or not citizens that belong to it should be vaccinated in their job place (nursing homes, for example).	None
Job	Defines a job by its name.	None
Citizen	Defines a citizen by their citizenship card number, birth date, name and gender. His address is also defined, as well as his job and the group he belongs to (might not belong to any group).	None
Address	Defines the address of a given place with its zip-code, street name, door number and county.	(1) a) door number ≥ 1

County	Defines a county by its name.	(2) a) name must be unique for a specific District
District	Defines a district by its name.	(4) a) name must be unique

3. Revised Conceptual Modeling

3.1 UML Diagram



3.2 Extra Restrictions after Revision

Name	Definition	Restrictions
Pathology - VaccinationGroup	Association that defines the pathology a vaccination group is made to neutralize.	(12) a) the pathology must be vaccinable (there must exist an association between a vaccine and this pathology)

4. Relational Model Definition

In all the explicit relations, the primary key is defined with an underline.

district(id, name)

county(id, name, district_id)
district_id : FK(district)

zip_code[1](id, zip_code, county_id)
county_id : FK(county)

address(id, zip_code_id, street_name, door_number)
zip_code_id : FK(zip_code)

job_group(id, name, vaccinated_on_job)

job(id, name, group_id, address_id)
group_id : FK(job_group)
address_id : FK(address)

pathology(id, scientific_name, common_name)

vaccine(id, name, producer, minimum_temperature, maximum_temperature,
prevents_pathology_id, inoculations_number, route, additional_info)
prevents_pathology_id : FK(pathology)

vaccination_group(id, minimum_age, maximum_age, priority_level)

pathology_reacts_adversely_to_vaccine(vaccine_id, pathology_id)
vaccine_id : FK(vaccine)
pathology_id : FK(pathology)

[1]The zip-code table was added onto the relational model in order to make the address table have all its FDs in the BCNF.

citizen(id, citizen_card_number, birth_date, name, gender, job_id, address_id)
job_id : FK(job)
address_id : FK(address)

job_group_vaccination_group(job_group_id, vaccination_group_id)
job_group_id : FK(job_group)
vaccination_group_id : FK(vaccination_group)

pathology_vaccination_group(pathology_id, vaccination_group_id)
pathology_id : FK(pathology)
vaccination_group_id : FK(vaccination_group)

vaccination_group_vaccine(vaccination_group_id, vaccine_id)
vaccination_group_id : FK(vaccination_group)
vaccine_id : FK(vaccine)

citizen_has_pathology(citizen_id, pathology_id)
citizen_id : FK(citizen)
pathology_id : FK(pathology)

citizen_belongs_to_vaccination_group(citizen_id, vaccination_group_id)
citizen_id : FK(citizen)
vaccination_group_id : FK(vaccination_group)

infrastructure(id, address_id, total_stored_vaccines)
address_id : FK(address)

storehouse(infrastructure_id, maximum_capacity, minimum_temperature,
maximum_temperature)
infrastructure_id : FK(infrastructure)

distribution_centre(infrastructure_id)
infrastructure_id : FK(infrastructure)

vaccination_centre(infrastructure_id)
infrastructure_id : FK(infrastructure)

inoculation(id, inoculation, number, date, vaccination_centre_id, vaccine_id, citizen_id)
vaccination_centre_id : FK(vaccination_centre)
vaccine_id : FK(vaccine)
citizen_id : FK(citizen)

delivery(id, distribution_centre_id, vaccine_id, amount, arrival_date)
distribution_centre_id : FK(distribution_centre)
vaccine_id : FK(vaccine)

transportation(id, shipment_date, arrival_date, amount, from, to, vaccine_id)
from : FK(infrastructure)
to : FK(infrastructure)
vaccine_id : FK(vaccine)

vaccine_storage(vaccine_id, infrastructure_id, amount)
vaccine_id : FK(vaccine)
infrastructure_id : FK(infrastructure)

5. Functional Dependencies Analysis and Normal Forms

The following table presents all **non-trivial** functional dependencies.

Most of the relations that come from an association between two other relations have no other attributes than those derived directly from the associated tables, thus have no functional dependencies apart from the trivial ones. Consequently, these relations display no functional dependencies in this table and are indicated with *None*.

Since every attribute of every table corresponds to only one atomic type, one can conclude that all the relations follow the First Normal Form. Simultaneously, given that all non-prime-attributes never depend on a subset of a relation's key, the Second Normal Form is verified. On top of that, since every relation holds dependencies of prime attributes or dependencies on attributes that constitute a key, the Third Normal Form is also verified.

Given the lack of real-world unique attributes, some relations lay only on one functional dependency, where {id} → {everything else}, being therefore on BCNF, and indicated with *None* on its *Analysis*. With this, when this is not the case, the *Analysis* column provides an explanation of the keys, as well as the clarification of why it still holds the BCNF.

In each functional dependency, its left side represents the keys of the given relation.

Relation	Functional Dependencies	Analysis
district	{id} → {name} {name} → {id}	The district's name is a natural key
county	{id} → {name, district_id} {name, district_id} → {id}	There can only exist one county per district
zip_code	{id} → {zip_code, county_id} {zip_code} → {id, county_id}	The zip-code is a natural key and the county can be extrapolated from it
address	{id} → {zip_code, street_name, door_number} {zip_code, street_name, door_number} → {id}	There can only exist one address with the same door number, in the same street, in the same zip code area

job_group	{id} → {name, vaccinated_on_job} {name} → {id, vaccinated_on_job}	The job group's name is a natural key
job	{id} → {name, group_id, address_id} {name, group_id, address_id} → {id}	There can only exist one job name per group in the same address
pathology	{id} → {scientific_name, common_name} {scientific_name} → {id, common_name}	The pathology's scientific name is a natural key
vaccine	{id} → {name, producer, inoculations_number, minimum_temperature, maximum_temperature} {name} → {id, producer, inoculations_number, minimum_temperature, maximum_temperature}	The vaccine's name is a natural key
vaccination_group	{id} → {minimum_age, maximum_age, priority}	None
pathology_reacts_adversely_to_vaccine	None	None
citizen	{id} → {citizen_card_number, birth_date, name, gender, job_id, address_id} {citizen_card_number} → {id, birth_date, name, gender, job_id, address_id}	The citizen's card number is a natural key
job_group_vaccination_group	None	None
pathology_vaccination_group	None	None
vaccination_group_vaccine	None	None
citizen_has_pathologoy	None	None
citizen_belongs_to_vaccination_group	None	None
infrastructure	{id} → {address_id, total_stored_vaccines}	None

storehouse	{infrastructure_id} → {maximum_capacity, minimum_temperature, maximum_temperature}	None
distribution_centre	None	None
vaccination_centre	None	None
inoculation	{id} → {inoculation_number, date, vaccination_centre_id, vaccine_id, citizen_id} {inoculation_number, date, vaccine_id, citizen_id} → {id, vaccination_centre_id}	A citizen can only take one vaccine shot in a day, and the vaccination centre can be extrapolated from it
delivery	{id} → {distribution_center_id, vaccine_id, amount, arrival_date}	None
transportation	{id} → {shipment_date, arrival_date, amount, from, to, vaccine}	None
vaccine_storage	{vaccine_id, infrastructure_id} → {amount}	Given an infrastructure and a vaccine, the amount of shots of the vaccine is unique and can be extrapolated from it

6. Restrictions

All restrictions that do not need the use of triggers were implemented and are listed bellow, according to each type of restriction.

6.1 Key Restrictions: Primary Key or Unique

Every table that does not come from an association has a parameter `id`. As such, all of them have a primary key constraint.

The association tables have a composite **primary key**, formed by the two `id`s of the tables that define the association. For example, pathology_reacts_adversely_to_vaccine has a composite primary key formed by vaccine_id and pathology_id.

There are several **unique** constraints. All of them are listed below.

- A district name must be unique;
- There can only be one unique county name for a given district;
- A zip code must be unique;
- An address, which is composed by a zip code, street name and door number, must be unique;
- A job-group name must be unique;
- A job, which is composed by its name, job-group and address, must be unique;
- A pathology's scientific name must be unique;
- A vaccine's name must be unique;
- A citizen's card number must be unique;
- An inoculation, which is defined by its number, date, vaccine and citizen, must be unique;

6.2 Referential Integrity Restrictions: Foreign Keys

Whenever there exists a parameter on a given table that references another table, a **foreign key** constraint is used. For example, in pathology_reacts_adversely_to_vaccine, both vaccine_id and pathology_id are foreign keys. Therefore, a constraint was set for both parameters. The aforementioned example can be applied to all foreign keys that were explicitly described in chapter 4.

6.3 Context Restrictions: Check

There are several check constraints. All of them are listed below.

- An address either has no door number or it must be greater than or equal to 1;
- A vaccine must require at least one inoculation;
- A vaccine's minimum temperature must be lower than or equal to its maximum temperature;
- A vaccination-group's age range must respect all the following conditions:
  1. the minimum age must be greater than or equal to 0;
  2. either the minimum age is lower than or equal to the maximum age or there is no maximum age;
- A vaccination's group priority level must be greater than or equal to 0;
- The number of an inoculation must be greater or equal to 1;
- The number of vaccines stored in an infrastructure must be greater than or equal to 0.
- A storehouse either has no defined maximum capacity or it is greater than 0.
- A storehouse's temperature range must respect one of the following conditions:
  1. there's no minimum temperature;
  2. there's no maximum temperature;
  3. the minimum temperature must be lower than or equal to the maximum temperature;
- A delivery must deliver at least one vaccine.
- A transportation must carry at least one vaccine.
- The dates of a transportation must respect one of the following conditions:
  1. there's no shipment date;
  2. there's no arrival date;
  3. the arrival date is greater than or equal to the shipment date;
- An infrastructure either has no information about the amount of vaccines it stores or it stores a number greater than or equal to 0.

6.4 Mandatory Parameter Restrictions: Not Null

Whenever a parameter is critical to a table and must exist, a **not null** constraint is set (except when it already is implied, *e.g.*, all primary keys). All of the parameters that are listed below cannot be null.

Table	**Parameter**
district	name
county	name, district_id
zip_code	zip_code, county_id
address	street_name
job_group	name, vaccinated_on_job
job	name
pathology	scientific_name
vaccine	name, producer, minimum_temperature, maximum_temperature, prevents_pathology_id, inoculations_number
vaccination_group	minimum_age, priority_level
citizen	citizen_card_number, name, birth_date, gender
inoculation	inoculation_number, vaccine_id, citizen_id
infrastructure	address_id
delivery	distribution_centre_id, vaccine_id, amount
transportation	amount, to, vaccine_id

7. Queries

The following queries tackle common questions one might have while managing the database at hand. Queries 2, 3, 5, 10, 11, 13, 14, 16, 19 and 20 are the ones that show a greater variety of SQL operands and operations and should be evaluated (according to the set limit), thus being explicitly marked with a '•'.

All other executed queries are present, however, as they are still relevant to the problem.

Query 1

```
Who are the citizens that belong to a given vaccination group?
```

Gets the list of citizens that belong to a defined vaccination group, being that group defined by the user (in the examples, the chosen vaccination group has ID = 1).

- ***Query 2***

```
What is the most common pathology?
```

Defines the pathology with the most infections of all pathologies in the database. The edge case of two pathologies with the same maximum value was taken care of, assuring that both appear in the result. This query was executed avoiding the use of `MAX`, as requested.

- ***Query 3***

```
What is the percentage of citizens that took at least one dose and are fully
dosed for all pathologies?
```

The first view `citizen_vaccine_numbers` counts the number of inoculations taken, the remaining, and the required per vaccine, per citizen. It also carries the ID of the pathology that is prevented with the vaccine.

Given that it is possible for a citizen to take more than one vaccine for the same pathology, the second view, `citizens_vaccine_pathologies`, groups the last view by pathology, returning as well the number of inoculations taken and inoculations remaining for the vaccine that has the lower number on this last parameter. This is, if a citizen took all required doses for the Pfizer® vaccine,

and just one dose for the Moderna® one, it would state that the citizen has 0 inoculations remaining for COVID-19, since they are already fully vaccinated with one of the vaccines.

With the results of this last view, the number of citizens that have 0 inoculations remaining per pathology is counted. This result is stated in view `fully_vaccinated_pathology`.

Since the second view already groups inoculations per citizen and pathology, if this result is grouped only per inoculation, the count of rows associated with each one corresponds to the number of citizens that already took at least one dose of one of the vaccines associated with each pathology. This result is stated in view `at_least_one_vaccinated_pathology`.

Then, the query that returns the real results starts by counting the number of people in the database. It then joins each pathology with the results of the last two views, in order to divide the count result with the number of citizens, thus getting the percentage for each of the required fields, these are the percentage of citizens that took at least one dose of one of the vaccines for the correspondent pathology, and the percentage of citizens that already are fully dosed for at least one of the vaccines of the correspondent pathology. It uses left joins so that pathologies that do not have either vaccines or inoculations can be shown with 0% as its result.

Query 4

```
How many doses are administrated for a given pathology?
```

Counts the number of inoculations of vaccines that prevent a given pathology. In this case, the pathology number 56 was used, which corresponds to COVID-19.

- ***Query 5***

```
How many doses of a given pathology's vaccines were administrated to a given citizen,
and how many are left?
```

For a given pathology (in this case, the chosen one was COVID-19), this query outputs the current vaccination status for all vaccines associated to the chosen pathologies for a user chosen citizen. In addition to this, it also displays the date of the latest inoculation for each of the vaccines (the output is also customized in case no inoculations of a vaccine were taken).

Query 6

```
What is the disease with the highest vaccination rate?
```

This query outputs the pathology with the highest percentage of inoculations per citizen.

Query 7

```
How many vaccines does a pathology have?
```

Counts the number of vaccines that prevent a given pathology.

Query 8

```
What are the vaccines for a pathology?
```

Gets all vaccines, returning as well the common and scientific name of the pathology that they prevent.

Query 9

```
What are the storehouses above 50% of its capacity?
```

Since the capacity of the storehouses is stored in the `Storehouse` table, and the current number of vaccines is stored in the `Infrastructure` table, it joins both tables, returning the ones where the number of stored vaccines divided by the capacity of the infrastructure is above 0.5.

- ***Query 10***

```
What is the percentage of people per county that are vaccinated for a given
disease (in this case, COVID-19)?
```

The first view `counties_with_covid_inoculated` gets the needed information for all counties with (in this case) COVID inoculated: how many people are fully vaccinated and how many are mid vaccination.

The intention, however, is to show the percentage of people **per county** that are vaccinated. As such, a second view was made to join the counties present in the previous view with the ones that have no people inoculated, and to associate the population to the corresponding county.

With this second view, a simple query that divides the vaccinated by the total population per county outputs the pretended result.

- ***Query 11***

```
What is the incidence per 100k of infection of a given disease (in this
case, COVID-19)?
```

Gets the number of citizens in the database, as well as the number of infected by COVID-19. Proceeds then to do the required calculation.

Query 12

```
What is the incidence per 100k per county of infection of a given disease
(in this case, COVID-19)?
```

Gets the number of citizens in the database, as well as the number of infected by COVID-19. Proceeds then to do the required calculation per county. (reutilizes the view of query 10)

- ***Query 13***

```
What is the percentage of people with a pathology by job group?
```

Creates two separate views: one that defines the number of infected citizens in each job group, and another that defines the number of people in each job group. Then, a simple query to calculate the pretended percentage is executed.

- ***Query 14***

```
What is the number of inoculations administrated per day for all
pathologies?
```

Joins the pathologies, the vaccines that prevent them, and the inoculation of these vaccines, grouping it by pathology and calculating the difference between the last and the first date in these inoculations. A left join is used so that in the case the pathology does not have vaccines, or the vaccines do not have inoculations already, these values stay null. The number of inoculations for each pathology is then calculated and divided by this date difference, resulting in the number of inoculations administrated per day, for each pathology.

Query 15

```
How many inoculations were administrated for a given pathology (in this
case, COVID-19) in a given day (in this case, 2021-03-05)?
```

Gets the number of inoculations where the day is the one mentioned, and the vaccine prevents the pathology mentioned.

Query 16

```
What is the capacity per capita to hold vaccines in each district?
```

Like previous queries, a view that displays the districts with capacity is created first, to then be created a capacity per district temporary table in whilst generating the demanded query. Each district's capacity value is then divided by its population, which outputs the desired result.

Query 17

```
What are the pathologies that have no vaccine?
```

Outputs all pathologies that do not have any associated vaccine.

- ***Query 18***

```
Which vaccines do not have a storehouse that could hold it (according to
temperature restrictions)?
```

Since vaccines have diverse and particular storage conditions, not all storehouses can hold a vaccine most of the time. There might exist, indeed, vaccines that do not have any storehouse that can store them safely regarding its temperature restrictions.

As such, this query outputs all vaccines which have a [min,max] temperature interval that does not fit inside any of the available storehouses in the database, by comparing both its max and min values (and also takes into account the fact that storehouses might not have a minimum/maximum temperature limit - which is NULL).

- ***Query 19***

```
What is the number of citizens by age group?
```

A query targeting the database demographic, which groups all citizens in age groups of 10 years and outputs how many are present in each of them.

- ***Query 20***

```
What is the percentage of citizens vaccinated with at least one inoculation
for a pathology by age group?
```

The first and second view are related. The second view, `vaccinated`, represents all the id's of all citizens that have been vaccinated with any COVID-19 vaccine. The third and last view, calculates the age group of all citizens calculating its age in years and then casting it to its decade.

With these views, the `SELECT` calculates the percentage of people vaccinated by age group pretty-printing the data with the concat (||) operator.

8. Triggers

The following triggers seek to avoid several integrity problems the database might have and keep it coherent and consistent. Just like what happened in the queries above, more triggers than the ones defined were needed for some sense of control over the data. As such, the three chosen triggers for evaluation are also marked explicitly with a '•'.

8.1 Executed Triggers

Trigger 1

```
Prevent transportation from being held to a Distribution Centre
```

Before inserting a transportation, if the destination infrastructure is a distribution centre, aborts the insertion.

- ***Trigger 2***

```
Keep the storage of an infrastructure updated according to the transportation
associated with that infrastructure
```

After a transportation, it is mandatory to update the respective derived columns `total_stored_vaccines` in infrastructure and `amount` in vaccine_storage, since they are all related, in order to keep the database updated. As such, upon insertion in transportation, a new vaccine storage entry is created if one does not exist yet and the referenced values are dully updated (in conjunction with trigger 3, and the fact that the trigger is in itself a transaction, it is assured that the values aren't altered if, in fact, the transportation is not possible).

- ***Trigger 3***

```
Verify if there are enough vaccines in the origin infrastructure and if there
is space in the destination infrastructure for a given transportation to occur
```

Before inserting a new transportation, verifies if the origin infrastructure has the vaccine that will be transported. If it has, verifies if it has the required amount for transportation. Finally, it verifies if the destination infrastructure, being a storehouse, has the required capacity to hold the number of vaccines that are to be transported. If one of these constraints fail, it aborts the insertion.

- *Trigger 4*

```
Verify if the destination storehouse of new transportation meet the temperature
requirements of the vaccine that is transported
```

As explained before, it is possible that a storehouse might not have the temperature requirements to hold a given vaccine. This trigger checks, upon an insertion on transportation, if the interval of temperatures of the destination storehouse intersects the interval of temperatures of the vaccine to store. If it does not, the insertion is aborted.

Trigger 5

```
Verify if an inoculation number is in the range of the number of inoculations
of a given vaccine
```

Before inserting a new inoculation, verifies if its number is positive and if it is in the range of accepted doses for the vaccine. If this constraint fails, the insertion is aborted.

Trigger 6

```
Keep the storage of an infrastructure updated according to the delivery associated
with a distribution center
```

This trigger does a similar job of the one done by trigger 2, but instead of being triggered upon insertions on transportation, it is triggered upon insertions on delivery.

8.2 Other remarks

Other triggers could've been implemented to make the database more strict and less prone to inconsistent information.

The referenced restriction in "Extra Restrictions after Revision" is a great example, which would simply require a verification of the existance of a vaccine-pathology association upon inserting in pathology_vaccination_group.

Another pertinent trigger would be to decrease the vaccine amount by one when an inoculation is given, to assure a correct in-and-out flow (the current state of the database only assures the correct increase of vaccines, not the other way around).