

UNIVERSITÀ DEGLI STUDI DI MODENA E REGGIO EMILIA

Dipartimento di Scienze Fisiche, Informatiche e
Matematiche

Corso di Laurea in Informatica

SCALABLE DATA SCIENCE AND TECHNOLOGIES FOR COPY NUMBER VARIATIONS STUDIES

Relatore:

Chiar.mo Prof.ssa Federica Mandreoli

Candidato:

Valentino Pisi

Correlatori:

Chiar.mo Dott. Riccardo Martoglia

Chiar.mo Prof. Cristian Taccioli

Chiar.mo Dott.ssa. Chiara Vischioni

Anno Accademico 2019/2020

Index

1 Introduction	4
1.1 Role of the DNA in the pathologies and tumors insurgence	5
1.2 The Data Set	8
1.3 Objective of the thesis	10
2 State of the art	11
2.1 Full Text Search	11
2.2 Data Analysis	14
2.3 Investment status in the sector and forecast of needs	18
3 The case study	20
3.1 Database creation and client/server differentiation	20
3.2 Information management	25
3.3 Data representation	29
3.4 Full Text Search and experimentation: Data Analysis through Postgres	36
3.5 Scalable Data Science	42
4 Conclusions	47

1 Introduction

In the past decades, the genes that compose the human DNA, and the ones of all the other species were a mystery. The role of each of these genes in it were unknown because the cost of single DNA analysis was really expensive. Due to this problem, it was quite economically impossible to achieve enough data to perform research. But, in the last years, thanks to the advancement of the technologies, the cost for a complete DNA test went, from 10000\$ to 1000\$, allowing researchers and to human beings a new phase of awareness and medical progression.

The new goal, as a consequence of those new achievements, is to discover which are the cause of the insurgence, and how genes are correlated to one of the main threads for most of the organisms of the planet hearth, cancer.

The aim of our research thesis was to build a powerful tool & platform to help the researcher in the analysis of genomic data. We have started to collaborate with Padova's University, side by side with Professor Cristian Taccioli and Doctor Chiara Vischioni. They have explained to us their theory about the role that can have specific genes, which can be present in multiple copies, can have in the DNA. In particular, some of these, which are still unknown, may play a central role in the onset and protection from pathologies, leaving us the task of identifying them.

My role within this internship consists in the management and in the manipulation of the data contained within the platform database, implementing a Scalable Data Analysis strategy in this. On the other hand, my colleague Fabio Bove, works on the implementation of different descriptive analysis models and several Exploratory Data Analysis instruments, to visualize and interact with the dataset.

1.1 Role of the DNA in the pathologies and tumors insurgence

Since the discovery of the DNA structure, the scientific world has continuously studied this helicoidal texture as the key to the comprehension of the mystery behind the organisms (*Homo sapiens* included), pathologies, and tumors.

The DNA is a double helix molecule composed of two chains that give it this shape. Each of these stores the genetic instructions for the development, functioning, growth, and reproduction of all known organisms and many viruses.

The totality of this is called the *genome*, which is divided into two parts:

- Genes: or coding regions, that encode proteins.
- Noncoding DNA: component of the DNA of an organism that doesn't encode proteins.

We had to work with the first one, the smaller and most exciting component of our structure. It represents just 2% of the whole genome, while the rest has only a regulatory activity. Most of the actual studies are done on the coding part (genes). Before they learned of their function, they were considered garbage. Now that we know them better, we can say that inside they mainly contain:

- Transposable elements: they are pieces of DNA that can move within the genome and can cause disease. They represent 50% of a gene.
- Intron: they are one of the non-coding part, which is spliced away before the translation of an RNA into a protein. They occupy 20% of a gene.

A significant event that can bring a spectrum of different activities in an organism, and that is directly dependent on the genes is the mutation.

We can observe mainly two different types of mutations:

- Formation: there are some mutations that an organism can inherit from the parents (through gametes and germ cells), and there are some others that can be developed during its lifetime. This mutation can be urged mutagens, carcinogens, other environmental agents (UV), or errors that arise during DNA replications. If the variation affects a somatic cell, it will not be inherited.
- Prevention: in our system exists various mechanisms and other genes can prevent mutations that can results lethal. Genes like TP53 (a famous tumor-suppressor)

are activated when DNA damage occurs. To avoid this mutation, it interrupts and destroys the result of this process allowing the organism to function properly.

An organism will instead find itself exposed to a possible tumor insurgence if the gene that is going to change is an oncogene or an anti-oncogene (or tumor-suppressor). As for these two types of genes, there are as many types of mutations that can notch the previous balances:

- Upregulation (or gene overexpression): it is a transformation that leads the gene to be more productive. It is convenient if this mutation occurs on an anti-oncogene because it will be able to counteract more strongly the onset of a possible tumor.
- Downregulation (or gene down expression): unlike the previous one, can reduce the effectiveness of a gene. This transformation is convenient if it is on an oncogene since it limits its harmfulness.

Peto's Paradox is named after Richard Peto, an epidemiologist who noted the link between the cancer rate insurgence and the weight of an organism [1]. To experiment with his assumption, he decided to use mice and to expose them to carcinogens trying to observe the relation with the time of the exposure to these substances and the insurgence of the tumoral factor on them. Later, he added a last element to the equation: the body weight. This previous addition generated an important question, why the *Homo sapiens* live 30 times longer than a mouse with a thousand times more cells?

"This question gave birth to the paradox from which it takes its name. Referring to the mutations previously described, the why of it is how the cancer is generated is:

Every time a human cell divides, it must copy its six billion base pairs of DNAs, and it inevitably makes some mistakes. These mistakes are called somatic mutation." [1].

If each cell has the same probability of undergoing a mutation, having an organism that has a more significant number of cells because it is larger, the probability of undergoing mutation should be higher as the size of the animal increases. Part of the explanation that we know from Peto's paradox derives from the elephant: redundancy of the copies of one of the most famous anti-oncogene genes in its genome, TP53, shows us how the high number of cells in an organism does not directly lead to a predisposition to cancer. These, however, are not the only characteristics, to the detriment of the paradox, which has emerged over the years of studies.

Peto's Paradox

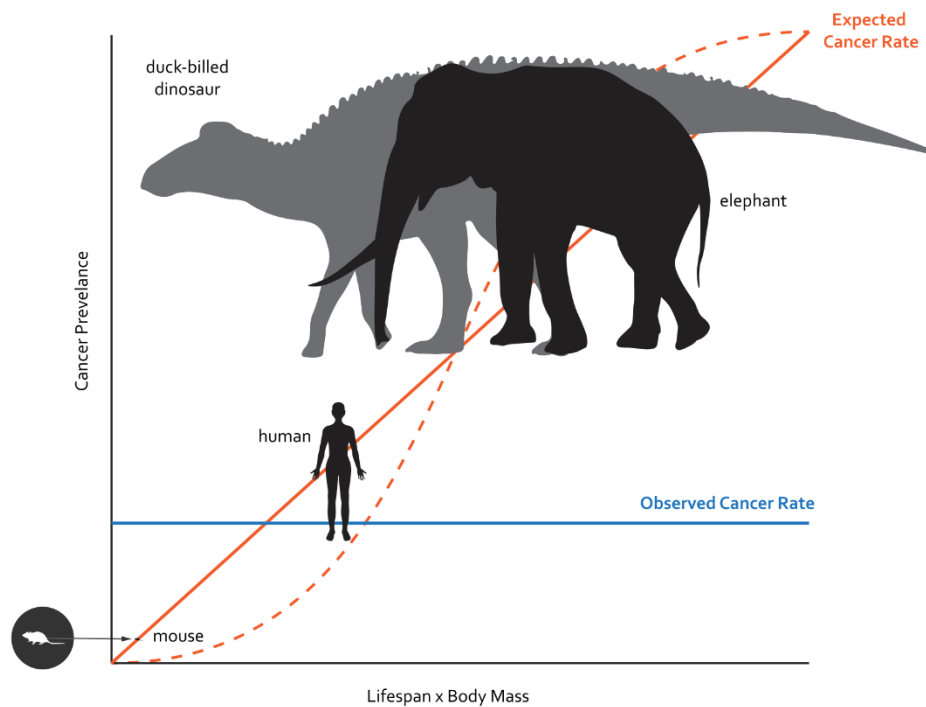


Figure 1 The relation between the cancer insurgence and the dimension of the dimension of an organism [1]

Another response that can be seen from the whale's lifespan, and which is related to cancer, is metabolism. A fast metabolism implies an equally rapid cellular reproduction. As mentioned above, the onset of a mutation with tumor consequences often occurs during the division of a cell due to an error in copying it. For this reason, an organism with a fast metabolism and an exponentially more significant number of cell divisions is condemned to high rates of cancer outbreaks. Obviously, in all categories, there are exceptions; one of these is the *Nanospallax galili*, a rodent that, due to a mutation that led to the over-expression of the TP53 gene, is practically tumor-free, unlike the other rodents and species with similar dimensions and consumption.

These are just some of the answers that can be deduced from in-depth analyzes of these organisms and do not preclude the discovery of further factors and roles that are still unknown to us.

1.2 The Data Set

A data set is a collection of data. It's an ensemble of information in the relational form splitting the data in tables, like in a database, where each column represents a different variable, while every row is a different member of the data set.

This type of collection, of separate sets of information, has become fundamental in every working and administrative sector, speeding up and making access to the information it contains safer. An example of declination that we can analyze, to understand how much is fundamental a data set structure for our society is deeply leaned to this system, is the environment of the healthcare. In their data set, every patient is profiled, saving his basic parameters, previous diseases, and where and how many times he was interned. The natural consequence of that is easier work for the doctors and a correct prognosis and timely response to the patient's needs and life expectancy.

The data that Padova's university has collected were divided into two tables: one containing 192 different species with, for each of them, their genomic combination in terms of CNVs (number of copies for each gene), and the other one, containing more specific data. The information stored in the last one was related to only 24 species on which there was enough medical information concerning: longevity, metabolism, weight, and the most precious data, the cancer insurgence rate.

In addition to that four fields, the Padova's researcher, Professor Taccioli and his collaborator, applied a double calculation of variance and, resulting from the following formula:

$$\text{copy_number_all_gene} > (\text{variance}(x)) / 4$$
$$\text{where } x = \text{copy_number_all_gene} > \text{variance}(\text{copy_number_all_gene})$$

The result of this query is 4 copy numbers that is used as a threshold to collect for each one of the 24 species, the 200 most present genes with copies greater than the threshold.

This table is the product of the information observed from a tremendous amount of necropsy performed on the organisms that are simpler to study and, in part, closer to human society.

Another information that came out during the feasibility study was the implementation, in the data set, of the known and relevant role, of some of our genes, in the onset of protection from cancer insurgence. To accomplish that goal, we have downloaded the gene's family, correlated to these two crucial functions, from geneontology.org to enrich the information that a single gene can retrieve to us.

Gene Ontology is a bio-informatic initiative whose aim is to unify and give a unique representation of a gene among all species. Among the other missions, they also have the task of making these data, and all those relating to this sector, equally machine-readable so that they can provide a single version of these universal data to anyone.

Thanks to this addition to the data set, we could create models that can identify patterns of the combination of the families that can justify, beyond the copy numbers, the cancer rate of some species and help to recognize that in the organisms of the first table.

With the combination of all these data that have been collected over the years, it was possible for us to build a database to be used as the foundation of our platform, and that contained all the data necessary for the structuring of models above them.

The total amount of data for each table is:

- 9996 tuples for the families' table.
- 192 tuples for the Species' table.
- 21036 tuples for the Genes' table.
- 3005109 tuples for the Copynumbers' table.
- 835088 tuples for the Classification's table.

The storage of all these records has led the database to request a space of 768 MB in memory, an affordable quantity thanks to the technologies and the size of the memory devices of these years.

1.3 Objective of the thesis

The overall objective of the thesis work was the creation of a web-based platform that allows researchers in biology and medicine to submit data analytics queries in an easy way, allowing them:

1. To pursue and verify mostly the role of some genes, with unknown or partially unknown function in the DNA, concerning cancer insurgence.
2. Take the new computed information and, primarily, estimate the onset of tumors in the other organism, which it was not possible to submit a thorough veterinary analysis to evaluate their death rate and to enhance cancer tools against this threat.

The web-based platform could, therefore, enable the introduction of novel tools such as a tool for the synthesis of new drugs for oncotherapy. This is by updating some of the existing drugs, still widely used by cancer patients, to target and neutralize those genes detected by our platform, which is contributing to the development of the tumor mass.

2 State of the art

2.1 Full Text Search

The Information Age (or Digital Age) is a historical period that begins in the early 20th century. This historical period is not yet finished and has found its birth since the last industrial revolution, and the delivery of an economy focused on information and data.

As mentioned earlier, this era has brought what can be said to be the key to industry 4.0 and today's society: digital data. They can be considered the primary weapon of recent years: they are used for customer profiling, in politics, in the economy, for military purposes, medicine, and many others. All these variations of use lead to two common problems:

- Retention: the first one is storage. All these data need to be stored and accessed; for this reason, there is a constant race to expand the size of the memories and their architecture.
- Research: for a sort of transitive property from the previous problem, with the continuous increase in the size of the data, it becomes even more challenging to find and recover the information necessary to respond to a possible request from the user.

To solve the latter mentioned problem, Full Text Search was designed and implemented. It's the technique on which most of the web is based. We can find it in Search engines (the google search bar) and inside some web pages for searches inside them. Each page (or document) is subject to an initial analysis of the content to proceed with subsequent indexing of this. This index will allow you to trace it and return it as a result, to the user who performed the search. The documents recovered by this technique will be represented by the title of this and a small extract, necessary to give an idea of the content to the user and to motivate the reason for its return and the index will allow applications to retrieve the required information efficiently and deliver it to the user who performed the search.

To give a definition, we can say: a full-text search is a search method that compares every word of the search request with all the words contained within a database or a collection

of documents. Web searches, programs for editing text, and many other applications are focused on this search mechanism. This technique is also extended to the individual operating systems to allow the user, through the structures on which this is based, to find the data indicated by his query quickly.

An index is a data structure, over the text, to speed up the research phase. It is particularly convenient when the document collection is large and semi-static; that is, the changes are scanned by time intervals and must not support thousands of character entries per second.

Not all strategies for implementing an index have the same efficiency, and this can vary, consequently, depending on the collection of data to be managed. For this reason, considerations must be made first:

- Search cost: depending on the adopted technique, a different strategy will be used for building the structure of the index. Research costs depend on this; each structure has its reading strategy, which will be useful for the collection to be managed.
- Space overhead: once the possible technique to be applied, we must take into consideration the need for storage space for the complete memorization of the index structures. This will be directly related to the size of the data collection that will be managed. The growth in the future of these data will also have to be considered.
- Cost of construction and updating for the indexing structures: it is the computational cost required for the initialization (before its creation) of the index and its maintenance and updating. It is highly dependent on the adopted technique. The computational cost can be understood as the calculation time necessary to carry out the actions mentioned above. It depends on the structures to be modified, the amount of data added or modified, and the access method deriving from the method used.

Without the necessary premises for choosing an indexing method, the main techniques for implementing a Full Text Search that we can find for managing a text database are:

- Inverted index
- Suffix arrays

- Signature files

The inverted index is a word-oriented system for indexing a text collection and still today is one of the most adopted strategies for the realization of many applications. For each term contained in the documents that are part of the collection, we will build a list of the documents that contain it. The inverted index will consist of:

- A vocabulary: this structure will provide the different words contained in a document without repetitions, and for each of them the corresponding document frequency, or the number of times we will find it in the text considered.
- Occurrences (posting list): for each word, a list will be stored containing the presence of it. This can be document-based, where it will store the documents that contain it and how many times (term frequency) or word-based, where instead it will go to take the documents that comprise it and the corresponding positions of the word inside it.

The space required is relatively small. For an instance of a vocabulary related to 1GB of text, this will occupy only 5 megabytes. This value can be further reduced through stemming and normalization techniques.

Occurrences need more space instead:

- The space required by a word-based inverted index will be $O(n)$ and will be around 40% of the text without stop words, and 80% if these are included.
- Instead, a document-based inverted index requires significantly less space. It will depend on the size of the document, requiring 20% to 40% of the text size as storage space.

2.2 Data Analysis

Collecting large amounts of information does not mean collecting them all. There is information that can be derived from these. For this reason, we are helped by an ad-hoc operation to be applied to our data that allows us to find this additional information and consequently enrich our tools.

Data analysis is a process applied to a set of data that allows you to inspect, clean, and transform them into models to discover useful information, conclusions, and help the decision-making phase.

There are many techniques and types of data analysis with as many names and sectors of use. Just think of the economic sector, the scientific sector, the social sector, and many others. This powerful tool helps users and companies that use it to make more thoughtful decisions and, at the same time, justified by further data that they previously did not have. Data mining is a data analysis technique which through statistical models and knowledge discovery seeks to provide information whose purpose is to predict rather than describe while, business intelligence, deals with the data analysis which strongly depends on the evaluation of multiple data aggregated among them, focusing mainly on the recovery of business information.

In statistical applications, data analysis can be divided into descriptive statistics, exploratory data analysis (EDA), and confirmatory data analysis (CDA). EDA focuses on discovering new features in the data, while CDA focuses on confirming or falsifying existing hypotheses. [2].

But implementing an analysis technique in a company or project also means considering multiple aspects. We need to evaluate the requests that we want to satisfy, the skills necessary for its implementation, the information to be treated, and the technological resources to be mobilized.

We can divide the actions of the Data Analysis process into some iterable steps:

- Data requirements: data is fundamental for the analysis. These, depending on the

context considered and the study to be achieved, vary in type and meaning. They can be numeric or textual, being combinations of these two types allowed and considered. These variables are related to the samples analyzed for the experiment (people, genes, market actions).

- Data collection: data are not always easy to obtain. Often, they must be requested and recovered from multiple sources. These can be data holders, sensors (such as cameras and thermometers), or to be acquired through interviews (market interviews), reading documents, or different online resources.
- Data processing: the data thus obtained must now be organized for the analysis phase. They are thus transformed into structured data, storing these in tables characterized by rows and columns.
- Data cleaning: once these have been tried, they could still present problems. These can be incompleteness, errors, or duplicate data. Cleaning and then removing these errors will prevent mistakes from occurring in the system. To avoid this, numerous data processes are applied, such as identification of inaccuracies, verification of data types, normalization, elimination of duplicates, and quality control.
- Exploratory data analysis: Once the data is cleaned, it can be analyzed. This can be done through a multitude of methods, referring to exploratory data analysis to start understanding the information contained in the set of data. This process may require further cleaning and data requests leading to an iteration of these actions. Usually, the transformation in visual form acquires a pivotal role to describe better the information and their relationships. This allows the messages obtained through this method to be made explicit and more comfortable for the user to read.
- Modeling and algorithms: models are applied; they are algorithms and mathematical formulas on the data to evaluate the connection among them and test the cause and effect relationship. Usually, a selected outcome is tested for prediction by also taking into account the error model performs in the prediction.
- Data product: is what is meant for final software. This will allow you to take data inputs and, through algorithms and models, to return the results deriving from them to the user.
- Communication: once the data have been processed through the system listed above, a technique belonging to the Data Analysis must be considered: Data

Visualization. This should help the end-user to acquire a better understanding of the obtained result and to navigate among the returned data. To achieve this goal, Data Visualization makes use of different types of graphs depending on the vital aspect to be highlighted and emphasized concerning the data on which it has been calculated.

In Figure, we can see how the points listed above related. These connections and iterations allow the operation of the data analysis technique.

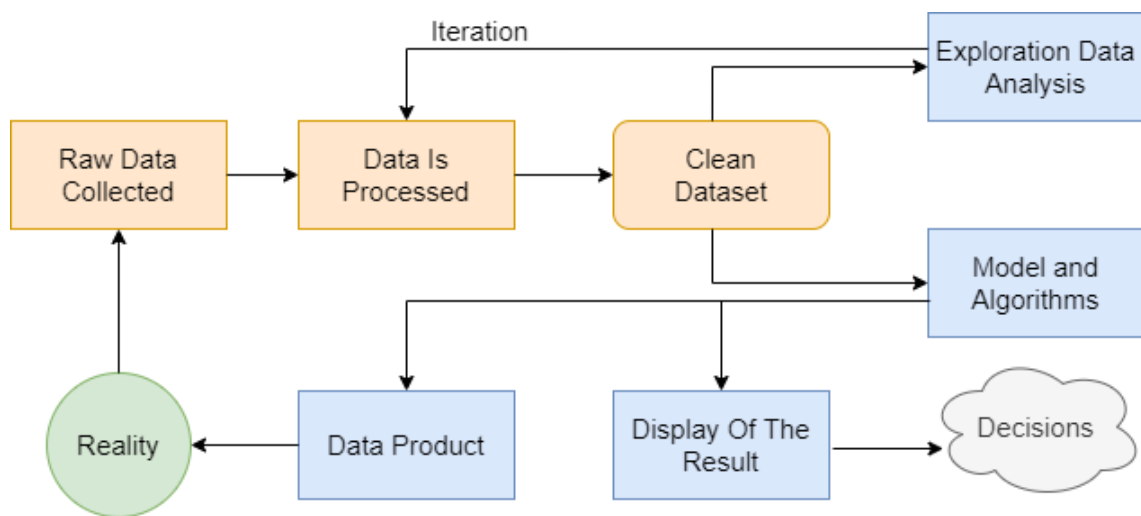


Figure 2 Operation diagram of the data analysis

To implement a data analysis tool, as it is often claimed, we don't have to start from scratch every time, building scripts for implementing the data analysis. For this purpose, there are already dedicated and prepared libraries for the addition of this technique within a project. For the creation of the platform, we used two of the most famous and appreciated Python libraries for their efficiency and support from the development teams: Pandas and Scikit-learn.

The first library has been mainly used for implementing different parts of the platform. Pandas is a library written using the Python programming language for manipulation and analysis. It was born for the economic sector but it has been made open source over the years and developed to be freely implemented in any type of project. This library provides facilities for the management of tabular data and time series.

The features that convinced us to use it, and that made it one of the most used libraries are:

- An object called DataFrame for data manipulation. This includes integrated data indexing.
- *“Tools for reading and writing data between in-memory data structures and different formats: CSV and text files, Microsoft Excel, SQL databases, and the fast HDF5 format”*. [3].
- An intelligent alignment of data and built-in management of missing data allowing to avoid numerous outbreaks of errors.
- *“Flexible reshaping and pivoting of data sets.”* [3].
- Columns can be inserted and deleted to modify the data pool to be manipulated.
- *“Aggregating or transforming data with a powerful group by engine allowing split-apply-combine operations on data sets.”*. [3].
- Integrated and optimized join and merge functions for high performance.

In addition to the characteristics listed above, DataFrames have integrated statistical functions that make it even faster and less expensive, from a computational point of view, to return information relating to the data stored in them.

The other library is scikit-learn. It is a free machine learning library for the Python programming language.

Scikit-learn options varied classification, regression, and algorithms to these related together with support vector machines, random forests, k-means, and many other structures. It has been designed to integrate with NumPy and SciPy numerical-scientific libraries for data management.

The integration of this library was limited to some functions for the calculation of the t-test and other statistical measures for the models. However, we have chosen to implement it to prepare the platform for the future addition of machine learning techniques that are widely supported by this library.

2.3 Investment status in the sector and forecast of needs

The continuous research in the medical and pharmaceutical field over the years has led to an increase in the life expectancy of man and pets. These developments have led the companies working in this sector to adopt investment policies in Research & Development and in the most advanced technologies to improve and increase the solutions they offer. Therefore, the equally growing IT sector comes to the aid of these realities. Quoting we can say: *“Large companies in general, these international businesses with subordinate operation centers, branches and/or factories (referred to as ‘business units’ or ‘BUs’ hereafter) located worldwide in particular, should take advantage of information systems (IS) to maintain, optimize and co-ordinate their operations and management. IT enables the provision of information services and collaboration of cross-department information dissemination to help multinational enterprises achieve value co-creation and service innovation in their operations and services”*. [4].

But not all companies can or want to invest in R&D because of the environment in which they are located. An example that we can give is that of Italy. It is one of the countries with the lowest number of patents and the least investing in this sector despite having major companies in it. The blame for this shortcoming can be attributed to pharmaceutical companies, which have a cost maintenance policy, and to an unfavorable context of control even if this, in recent years, is increasingly finding relaxation and encouragement from the government, bringing to an increase in investments in the sector. [5].

An example of a virtuous country in this field is Australia. In the 2006-2007 financial year, the Australian biotech sector reported a fundraiser of almost € 300 million from rights, drug development, device manufacturing companies, and individuals. This stimulus made Australia the first biotech hub in Asia-Pacific, bringing further investments from external areas, such as the UK, that provided them with management software, reinforcing with this technological tool further to strengthen the solidity of their sector and their leadership. [6].

We can deduct from these two examples how a decisive investment in this sector, and in the IT-related one, can lead to steady growth of the companies that are part of it, going to stimulate and speed up the creation of new medicines and patents that give sector itself an income over the years for future investments. We can also note that these are not homogeneous all over the world while still being able to appreciate a general increase in investments in the whole sector.

3 The case study

3.1 Database creation and client/server differentiation

The first step, and consequently, the first part that has been implemented in the project, is the database, where data will be stored.

A database is a collection of organized data that often take more advanced forms through modeling techniques. These data are stored on a computer, which makes the content available to authorized users employing certain security measures. This specific data structure is made possible through the implementation of DBMS software.

The database management system (DBMS) is the system that allows an entity to interact with a database to retrieve and analyze the data it contains. Also, the software provides the necessary tools to administer the database. The set of databases, the DBMS, and the applications connected to them are called "database systems."

To be able to give life to the platform and to implement this data structure, we have found the answer to all the features we need from the combined use of Django and PostgreSQL. Django is a high-level Python Web framework that allows developers to create web applications quickly and which integrates administrator interface for data manipulation. Its core is an MVC architecture composed of:

- Model: is an object object-relational mapper (ORM) that allows Python data models to interact and recover with the data contained in a Database.
- View: is a web template system that allows Django to process HTTP requests and to provide responses of the same type.
- Controller: is a regular-expressions-based URL dispatcher that allows the system to trace the requested resource and to activate the functions necessary for its construction and return.

Django also integrates tools for the implementation of databases through MySQL but, having evaluated the characteristics of this and comparing them with those of PostgreSQL, we opted for the latter. MySQL in its favor had the advantage of the high speed in reading large workloads at the expense of the competition while, PostgreSQL was created with more balanced performances, with very light readings (InnoDB) slower than MySQL but, with the advantage of being able to write large amounts of data with

proper management of competitions. The other differences that made me lean towards the latter were the orientation to objects, compared to the pure relational of MySQL, the strong predisposition to data analysis, the possibility of parallel queries, without read-lock, for multi-core systems and protection of the integrity of the database. All this matched the operational needs of the platform and the characteristics of the server hardware.

PostgreSQL is a powerful, object-relational database system that offers itself as an open-source alternative to RDMBS programs such as Oracle's MySQL. Over the years, it has acquired a strong reputation thanks to its recognized characteristics, such as architecture, reliability, data integrity, robust feature set, and extensibility. Thanks to the support of the open-source community, it can boast of a vast number of powerful add-ons.

To proceed with the first step of creating the data structure, I installed through the terminal PostgreSQL in the Server Operating System. The server, which was provided to us by the University of Modena and Reggio Emilia, is a machine with a 12-core processor, 32 GB of RAM, 2 TB of memory, and 64-bit Ubuntu as OS.

Once installed, we created the database named 'geni' with a relative password to limit access to it. PostgreSQL will make available the database with this name every time the machine is started. The second step to allow the population and the use of the structure by the platform was to connect our Django framework to PostgreSQL.

This was done by specifying the engine to be used in Django's settings.py file, under the heading 'DATABASE' as follows:

```
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql_psycopg2',
        'NAME': 'geni',
        'USER': 'geniutente',
        'PASSWORD': 'password',
        'HOST': 'localhost',
        'PORT': '',
    }
}
```

Figure 3 Data to be inserted in the settings.py file to connect the PostgreSQL database to Django

Once this operation is complete, only the third and final step is missing. It consists, as mentioned above, in setting up Django Models. These, specified in the `model.py` file, will be the models that will relate to the individual tables in the database. We had to create a model for each of them. Each model is composed of:

- Model name: with this method of forming the database, the model name will give its name the table it will create and consequently to which it will refer.
- Fields of the table: here, you must define the names of the fields of the table, to which the related model will connect. In addition to this main feature, the following will also be specified: the type of data that the field will contain and optionally, the preferences regarding the management of some events (insertion of a null value, empty value, default values).

After completing the preparatory settings for creating the database, by first using the Django migrate command, which store the changes made to the `model.py` file, and subsequently with the Django makemigrations command, we created the tables in the structure considered. Their composition will be the same as that of the models.

For the database population, multiple versions of the create function were written. This was due to the addition of optimizations and the change of platform hardware. These versions can be divided mainly into three:

1. Create – initial version: without optimizations, the only goal was proper functioning.
2. Create with bulk mechanism: the first update aimed at optimizing the computational cost on "client" type devices, devices with significantly lower hardware power than that of a server.
3. Create with bulk and thread mechanism: the second update aimed at optimizing the distribution of the calculation across multiple threads. As mentioned above, the machine provided by Unimore allows us to create up to 64 threads for parallelizing more critical parts of code. This version has been designed for "server" type devices, with very high hardware power.

The most significant differences between these three versions are the computation times necessary to complete all the required operations. This limit that concerns the version one can make the server unavailable for a long time.

As for the first technique, the tables were considered sequentially. First, the three main tables (Genes, Species, Families) were created sequentially and then, in the same way, the two tables born from the N to N relationships (Copynumbers, Classification). The slowness of this method lay in inserting records into tables. These were, at each cycle, entered one at a time into the database, consequently having to stop pending the writing operations of the RDBMS. The algorithm operated according to the following scheme:

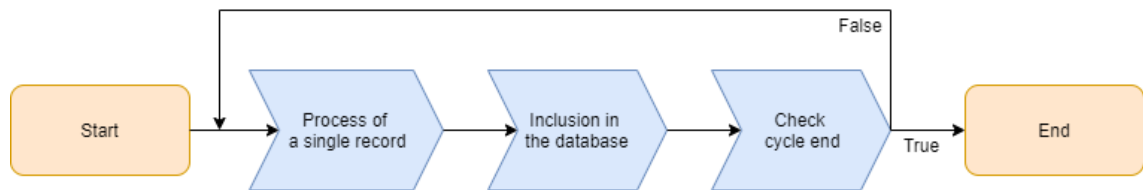


Figure 4 Calculation steps performed from the first version

The population of the entire database using this method took 12 hours, too much downtime, considering that our platform must always be accessible via the web.

The second technique is the first optimization of the create function. This improvement was achieved by adding bulk_insert (or bulk create). These are processes that allow us to load multiple lines into an RDBMS simultaneously. At each bulk, the whole table was passed; therefore, at each cycle, all the related tuples were built. All tables, as for the previous version, are created sequentially. The algorithm can be schematized as:

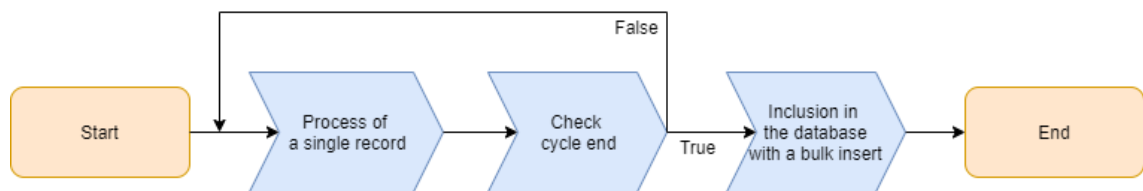


Figure 5 Calculation steps performed from the second version

The time required to perform all the processes of this version is 4/5h depending on the hardware characteristics of the host system. It was found that in systems with an SSD memory, the time required was 4 hours, while those equipped with HDD took an hour longer. This is the recommended version for obsolete hardware with dual-core processors.

The latest version is an upgrade of the second one, designed for hardware systems with

multi-core and more performing processors. Threads have been used to achieve this further optimization. Multiple threads can exist within a process; they can run concurrently and share resources such as memory. This implementation allowed the parallelization of the creation of the three main tables and, subsequently, of the two tables of the relations N to N. The algorithm can be seen in the following scheme:

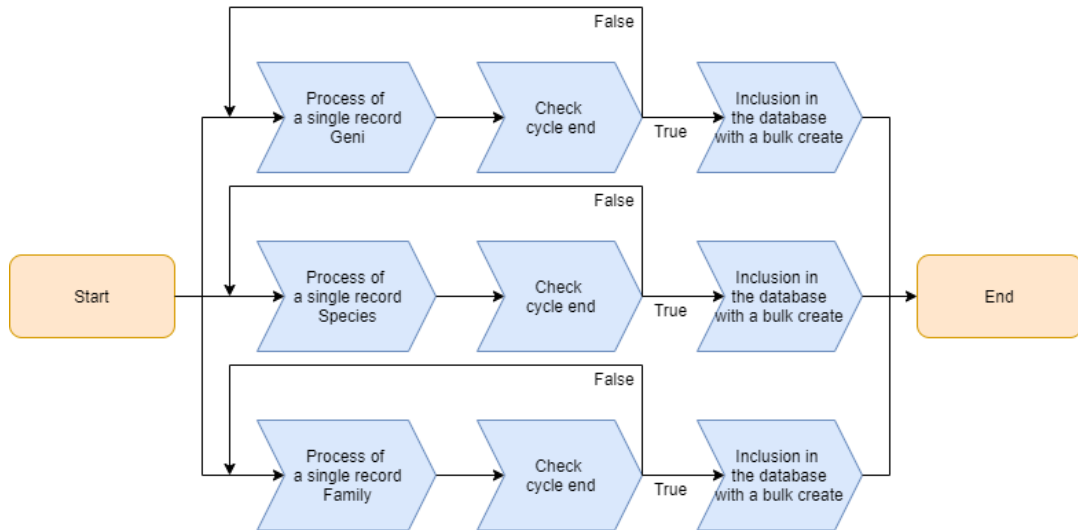


Figure 6 Calculation steps performed by the third and final version

The calculation time required, on the hardware we use, to complete the population of the tables is 1/2 hours. A timing more than acceptable given the amount of data but, this does not want to be a stop for future optimizations. It is a solution suitable for server systems. We are comparing, in the following graph, the times of the three versions developed during the internship. We can see how with "small" improvements, we have found enormous benefits in the timing.

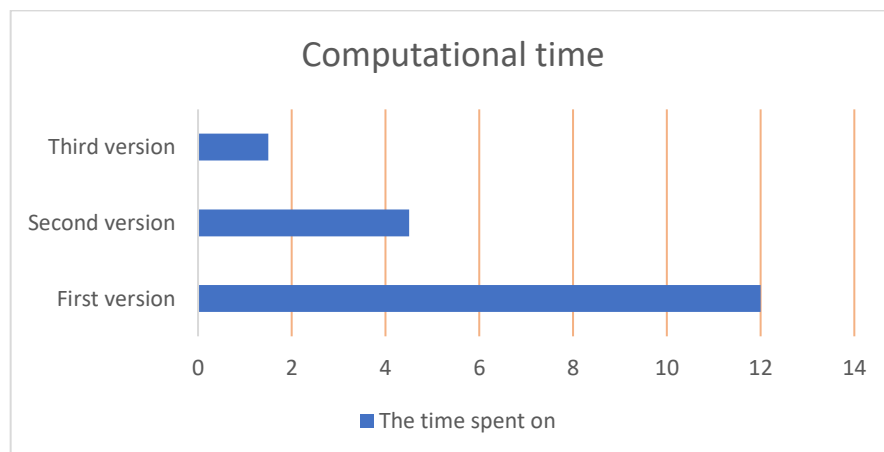


Figure 7 Graph that highlights the times of the three versions of the function create

3.2 Information management

As described in the previous chapters, we have relied on the data structure of the database to manage the information. Each table belonging to this structure represents a single component of our information set.

To this end, the first step was to analyze the data to store. The first information we had to store was related to the species. As described in the early chapters, the information pertaining to them has been retrieved from two different tables:

- The general table contains the names of all 192 species and all their genomes.
- The smaller and specific one contains the biological parameters of 24 species and their 200 most present genes.

Padova's researchers asked us to maintain two different tables but, after a careful evaluation of the relationships among these data and how they were selected, we decided that the two tables would give birth to the species table. So in order not to risk looking at only part of the information, these two tables gave birth to the Species table, in which the data relating to all 192 species were stored, leaving the biological data of the species not belonging to the small table to None.

Another fact that we have considered are genes. These were recovered from the large table. The information found for them is their scientific names and, thanks to the data collected online, whether they are oncogenes suppressors, oncogenes, or not. The combination of these allowed us to create an exhaustive Gene table composed of 21036 different genes.

The last of the three main tables that store the information that is fundamental for us is that relating to families. This information was not provided to us directly by Padua but was retrieved by Gene Ontology. We were able to recover them from it as this is one of the sites most used by researchers, given the reliability of its information. So, in the family table, we have stored the names of 9996 different families that could be related to nutrition or the onset of cancer.

The relationships between these tables are:

- A species has multiple genes, and a gene is in various species.
- A gene has numerous families, and a family is made up of multiple genes.

Having made these opportune premises, we can trace the related Entity-Relationship model:

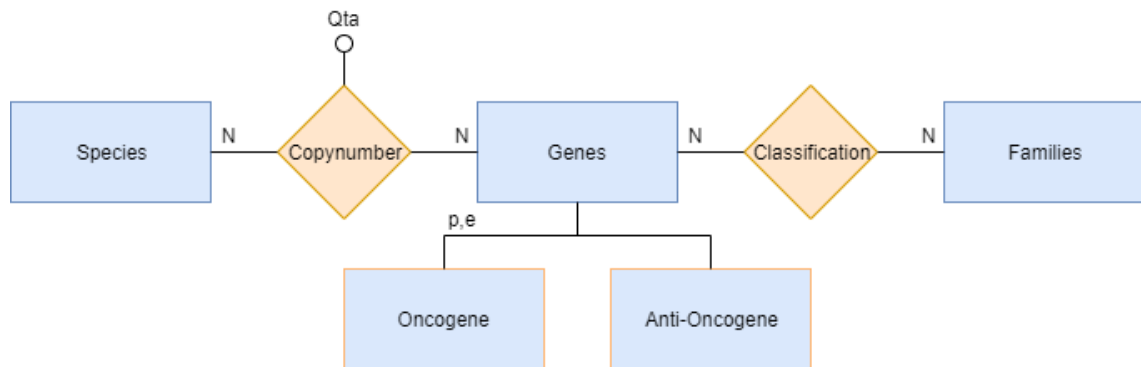


Figure 8 Entity/Relationship model of the database

From the following scheme, we can deduce the existence of the tables resulting from the relationships N to N. The first, Copynumber, containing the number of copies with which a gene makes up the genome of a species while the second, Classification, determines the families the genes belong to.

Now that we have defined the most general structure of our database, we can go to specify how the tables that we have just listed are individually composed by listing their fields:

- Species:
 - specie_name: Char Field, which stores the scientific name of the species in the database.
 - specie_longevity: Float Field, which contains the longevity of a species in years (Yrs).
 - specie_cancer: Float Field that stores the rate of onset of cancer in a species / 100 (5% = 0.05).
 - specie_metabolism: Float Field indicating the metabolism of a species. It is stored in Watts (W).
 - specie_avg_weight: Float Field, which records the average weight of a specific species. The value is to be understood in kilograms (Kg).

- Genes:
 - gene_name: Char Field, which stores the scientific and standardized names of genes.
 - gene_dd_tumor_suppressor: Boolean Field indicating whether the gene is an oncogene.
 - gene_dd_tumor_oncogene: Boolean Field indicating whether the gene is an anti-oncogene.
 - gene_reference_link: Char Field designed to link the page, related to the gene, of an external site at the researchers' discretion.
- Family:
 - family_name: Char Field containing the name of the genomic family.
- Copynumbers:
 - copy_number_qta: Integer Field indicating the number of copies of a gene in the genome of a species.
 - copy_number_gene: Foreign Key that connects the tuple to the gene, from the Genes table, to which it refers.
 - copy_number_specie: Foreign Key that connects the tuple to the organism, from the Species table, to which it relates.
- Classification:
 - classification_gene: Foreign Key linking a record to the gene, of the Genes table, to which it refers.
 - classification_family: Foreign Key linking a record to the family, of the Families table, to which it refers.

In addition to the information already explained, further specifications must be made. For each Float Field of the database, the null value has been made admissible (e.g., longevity for the species that are not present in the small table) and None has been set as default value. Further attention has been taken with the Foreign Key camps. Also, for them, the null values have been enabled and, besides, the automatic insertion of the null value has been set in case of cancellation of the tuple to which the field refers. To reduce the space needed by the database, and the size of the Copynumbers table, the links with copy_numbers_qta equal to 0 have been eliminated. This has allowed us to reduce the storage space from 4 million tuples to 3 million tuples, speeding up the time needed for

the execution of the create function. With the evolution of our platform in recent months, due to the new queries requested by researchers, we are considering reintegrating these million tuples to save computing time for building the answers for these queries.

By combining the E/R model and the data of each table, the final database schema is the following:

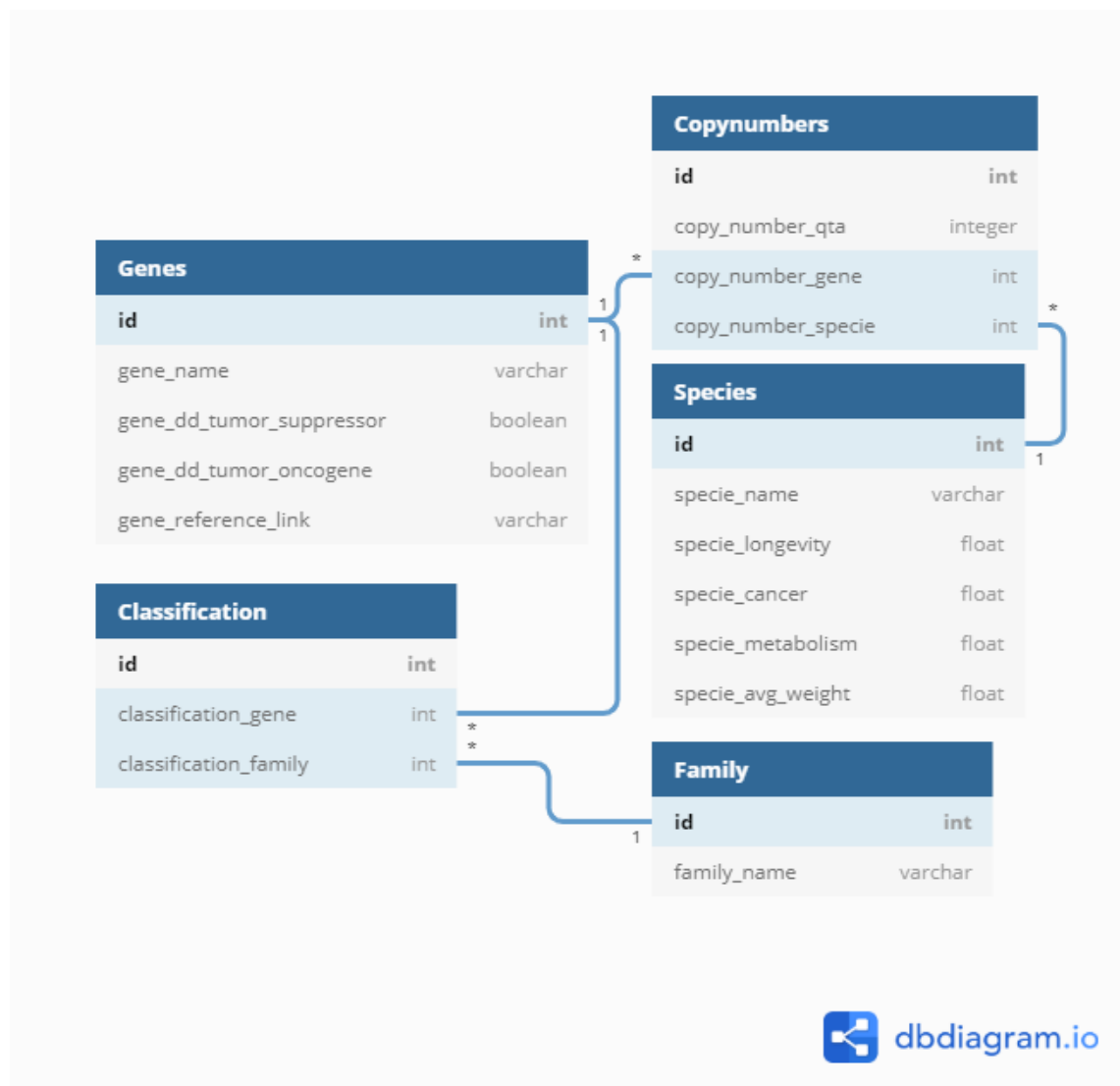


Figure 9 Database structure using the logic model

3.3 Data representation

A form is an interface created for users, which we can find inside a web application, that allows them to insert the data they want to send for insertion or query, from the client-side to the webserver. These interfaces are made up of different elements that allow a more detailed and, at the same time, guided and controlled insertion. The aspects that my colleague and I used for the creation of the forms, the fulcrum of our platform, were the text fields, and the drop-down menu.

Another explanation and premise that must be made are that of how to send the form containing the user data and the one with which to return the requested data. These are:

- GET: it is the most straightforward and most immediate method. It is recommended for those requests in which it is useful to store the URL with the parameters that have been passed to it to cache such data (e.g., storing a search). The downsides are the ability to read user data from this, and that specific browsers limit the length of URLs; this limitation reduces the amount of data that can be sent with this method. It is recommended for requests concerning small and few data.
- POST: this method differs from the GET method in the way data is sent. These are not passed through query strings and cannot be tracked even in server access logs. It is beneficial in case of the registration form, when many parameters must be passed or when sensitive data must be sent.

The GET method was chosen for receiving and sending the autocomplete requests and the POST method for sending and receiving form data.

What makes all input forms easy to use is the autocomplete function that was integrated into the interface we will discuss later. It appears as a menu below the text field to suggest the values allowed and available in the database. As an example: by starting to type the first letters of a gene, this function will indicate the early ten genes that begin with the ones entered. These results will gradually decrease as we insert more letters into the field until suggesting the gene we have searched for. This function aims to avoid typing errors and faster query insertion. All this has been possible, as previously said, through the GET method. As each letter is typed, the entire query that the user has typed at the moment

will be sent to the server, and, using the same method, this will return the first ten results (if available) that match the received string. This response will be handled by a JavaScript function, which will take care of reconstructing the suggestions menu.

Four different modules were introduced in as many pages: Home, Organism, Gene Exploration, and Analysis.

The Home page is the first page that a visitor will see when (s)he visits our platform; the purpose of our platform is presented in this page. Here the form consists of a single text field and a drop-down list to select the type of research: between genes or species.

Figure 10 Form of the Home page

By entering, for example, the name of a species within the field, after selecting this type from the dropdown, and clicking on the Execute button to send the query, a modal will open containing the answer to the request.

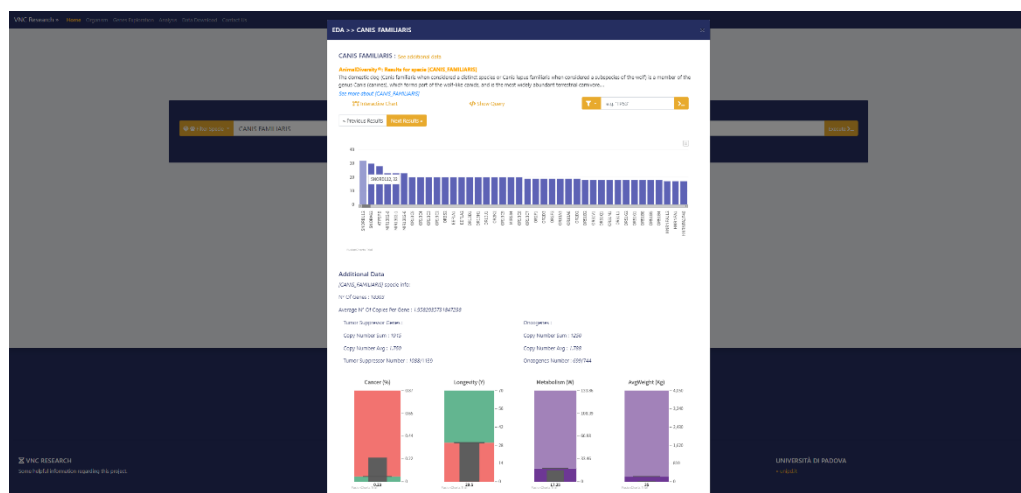


Figure 11 Modal of the response relative to the form of the Home page

The information contained within the modal of this page is part of the Exploratory Data Analysis model developed by my colleague Fabio Bove. Inside we can find a brief description of the organism sought and the graphs that go to show its biological parameters and its genome. The user will be able to interact with the chart and, by clicking on a gene, (s)he will be ready to relaunch a query on that gene going to deepen, as EDA supports, the search more and more in detail. For more information about it, I invite you to read his thesis called “Exploratory Data Analysis & Data Engineering Techniques for the study of Copy Number Variations” in which these aspects are treated and described in detail.

On the Organism page, searches focused on organisms are allowed. The form consists of 5 text fields: one for the name of the organism to be searched and the other four for the four biological parameters. These last are equipped with a drop-down list to select whether to search for organisms with a higher, lesser, or equal value than the one entered. If the name of the organism is entered, the fields relating to the biological parameters will be disabled.

Figure 12 Form of the Organisms page

The result of this query that we have considered will be a modal containing four boxes: one with the description of Wikipedia concerning the returned animal, the second containing the parameters of the selected organism (they will be None if this organism is not part of the small table), the third containing the list of its entire genome and the fourth with an interactive Sunburst type graph.

This will show the oncogenes and anti-oncogenes contained in the organism and, going to click on the ray relative to one of the genes, the EDA modal will open as for the Home page.

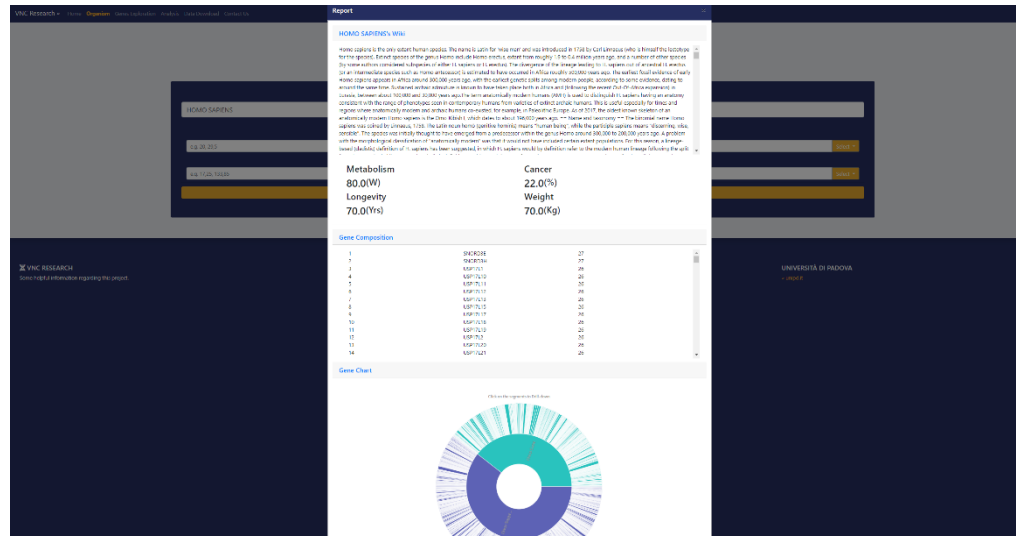


Figure 13 Modal of the response relative to the form of the Organisms page

If instead we leave the field related to the name of the species empty, and we go to specify one or more values in the fields related to the biological parameters, we will obtain a modal that will show, instead of the box related to the genome, the species that respect the constraints set by the user and in the box where before there were the parameters of the single animal, the parameters that the user-specified (those not indicated will remain null). The remaining boxes will remain unused.

For research concerning genes, the user can find the form dedicated to them on the Genes Exploration page. This form consists of two text fields and two drop-down menus. In the first text field, the user can specify the name of the gene to be searched while in the second, the number of copies of the gene to explore with the relative drop-down list to select whether the number must be higher, less or equal than the specified value. The last drop-down menu allows the user to choose whether to search for oncogenes or anti-oncogenes.

Figure 14 Form of the Exploration page

If we only enter the name of the gene we are looking for in the first text field, we will get a modal characterized by three boxes. The first will contain the Wikipedia definition of the gene sought; the second will indicate to the user all the organisms that comprise it and its copy number while, the last box, will contain a sunburst graph where each ray will show a different animal that contains the gene and in how many copies. This final information is the same as that of the second box but will allow the user to have a clearer idea of the distribution of the gene in the species of the database and, by clicking on a radius of the graph, the EDA model related to the indicated animal will be displayed.

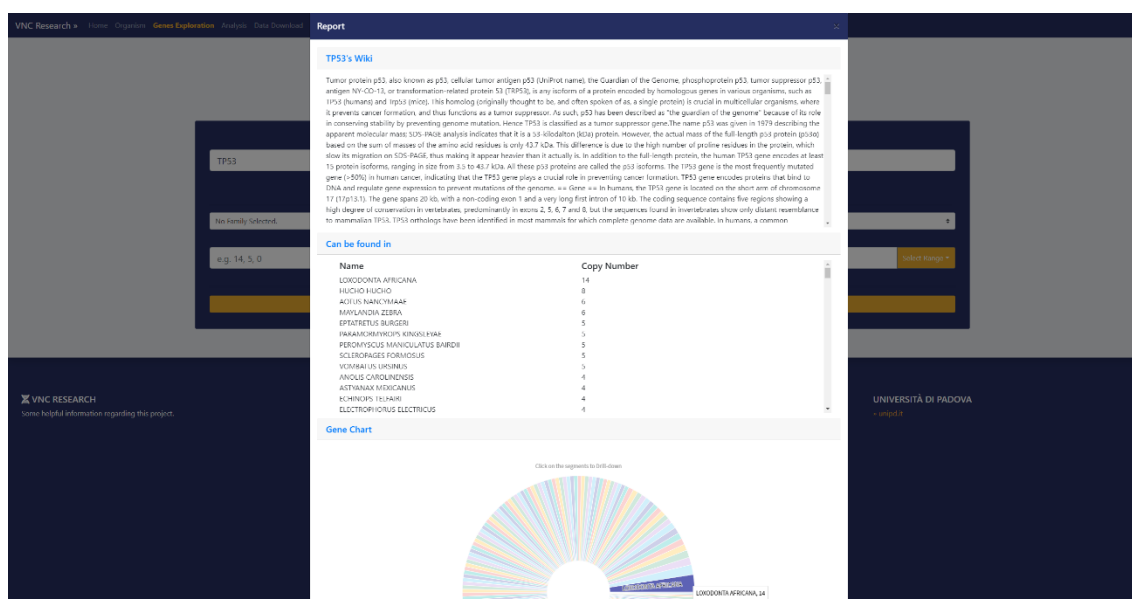


Figure 15 Modal of the response relative to the form of the Exploration page

This form supports multiple combinations of queries. If you specify, in addition to the name of the gene, the number of copies, the modal will return in the second box the species whose quantity of that gene respects the constraint inserted. If we remove the name of the gene and keep the number of copies, the modal will disable the first and third boxes while, in the second, it will indicate the genes and the relative copy numbers of these that respect the constraint inserted. The last two queries that are enabled are those related to the role of the gene. If we select the oncogenes from the drop-down menu, the modal will return the first and third disabled boxes, as before, and in the second, the genes that generate cancer onset and the maximum number of copies with which it can be found. If in combination with the menu, we are going to specify the constraint regarding the copies, we will obtain the oncogenes and the relative copy numbers that respect the entered threshold.

To execute the query, these forms have the same method. When the user fills out the form and presses the Execute key, the form sends the data entered by the user to the Python function, relative to the interface used. The first task of the function is to understand which combination of queries the user wants to execute and, checking which fields he has filled; this will build a RAW query in SQL. Once these checks are complete, a connection is opened with the RDBMS to run the query. Once the query result is received from the database, to speed up the calculation, the function will start two different threads. This will allow the platform to reduce the latency between request and response. The first thread deals with the construction of the structures necessary for printing data and retrieving information, concerning research, from external sites. The second thread is dedicated to the construction of the chart. The graphics used for these forms are created using the FusionChart libraries and, to draw them, the data must be sent to a JavaScript file in JSON format. Once the execution of the two threads is finished, we will send the information calculated by them to the JavaScript file, which takes care of receiving the JSON responses and publishing the data for the form concerned. We can, therefore, schematize the function responsible for the forms as:

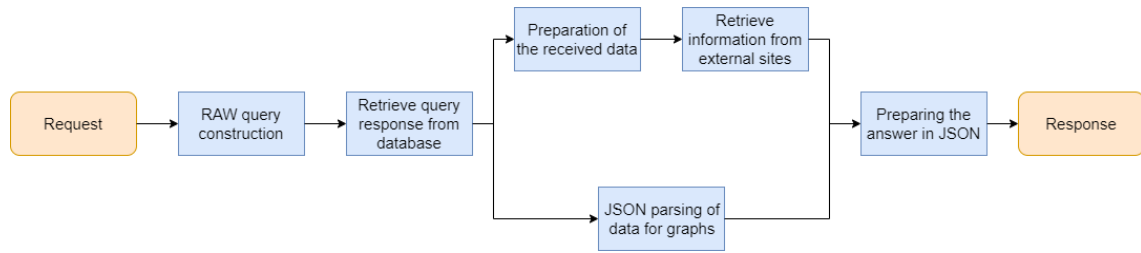


Figure 16 Calculation steps performed to return the user query result

Fusionchart is a library aimed at creating rapid charts and with integrated functions aimed at giving to the user statistical data while maintaining a relatively low computational cost. These libraries have been created for multiple programming languages and different integration methods and are widely used in leading companies in the IT sector. To further reduce the timing for the responses of our platform, we decided to integrate them into our Django framework.

As for the last form, this is the one relating to Full Text Search and will allow the user, given a syntax dedicated to it, to perform more complex queries. The dynamics concerning the functioning of this will be addressed in the next paragraph. As mentioned above, requests and responses are made in JSON.

JavaScript Object Notation is an open standard file format used for the exchange of information. It allows us to send objects having the attribute-value structure and arrays. This is a language that derives from JavaScript and is supported by numerous languages, such as Python, with integrated parsers. For this reason, the forms of our platform use this language, to allow data to be sent using a method common to all the languages we use, avoiding errors related to the type and format.

For each form we have mainly created:

- An HTML page for the interface structure.
- A Python function for receiving, searching, and sending data.
- A Javascript page that acts as a middleman between the web page and the Django server on which the Python scripts reside through JSON.

The information exchange among these modules is performed by means of the JSON language.

3.4 Full Text Search and experimentation: Data Analysis through Postgres

By connecting to the previous paragraph, the form related to full text search consists of a text field and a drop-down list to choose the search options. If we select the "Advanced Query" item from the aforementioned menu, we could enter the queries to be processed using this technique. In the following image, we can observe the form described above:

The screenshot displays the VNC Research website's 'Analysis' section. At the top, a navigation bar includes links for Home, Organism, Genes Exploration, Analysis (highlighted), Data Download, and Contact Us. The main heading is 'Analysis.' with a logo of orange dots. Below this is a dark blue box containing an 'Advanced - Data Analysis Query' form. The form has a 'Choose a Filter' dropdown, a text input field with the example query 'e.g. select: (gene_name) filter: [(specie_cancer < 0.5), (copy_number_qta > 5)]', and an 'Execute' button. Below the query field, there are links for 'Genes Quantity', 'Common Genes', and 'Genes Intersection'. Further down, there are links for 'Cancer Resistant Organisms', 'Cancer Prone Organisms', 'Oncogenes', and 'Tumor Suppressor Genes'. Below the dark blue box, the section 'Common Genes' is visible, with a subtitle 'Common Genes within a group of Organisms (e.g. Cancer Resistant, Cancer Prone...)'. A paragraph explains that a Multi-series Column chart is used to plot data for multiple datasets and to analyze and compare data points grouped in sub-categories.

Figure 17 Form of the Analysis page

Unlike the other forms, where the queries had limited combinations, and the possible values were suggested by mechanisms such as the autocomplete, here to give the user infinite combinations, the removals of these guides were necessary for the favor of a new dedicated syntax.

The simple syntax that has been created for the Full Text Search of our platform allows the user to request any column of any table in our database. To this freedom is also added the possibility of inserting unlimited constraints to allow to filter the information it requires meticulously. Therefore, the language is divided into the following three parts:

- **show:** within the brackets that follow this keyword, the user must specify the name of the fields containing the information he is looking for separated by a comma.
- **filter:** as for the previous keyword, the constraints relating to the parameters will be inserted within the limiters. To do this, the user will have to write the name

of the field and, if it is of the string type, he can select the name, without quotes, to search by using "=". For this field, you can specify infinite values using "&" as AND. If, on the other hand, the field to be filtered will be numeric, the user will be free to use combinations of the symbols higher ">", lower "<" and equal "=" to select the necessary thresholds. Filters must be separated by commas.

- `exclude`: it has the same structure as the keyword previously described but, the values resulting from the filters specified in these brackets will be subtracted from the values selected by the parameters indicated in those of the filter key.

To create such a powerful tool, requests to the database could be computationally prohibitive, bringing under the weight of multiple applications to the crash of our server. For this reason, I chose to use Pandas, for information management, and to recall the concept of cache memory.

“Cache memories are used in modern, medium and high-speed CPUs to hold temporarily those portions of the contents of main memory which are (believed to be) currently in use. Since instructions and data in cache memories can usually be referenced in 10 to 25 percent of the time required to access main memory, cache memories permit the execution rate of the machine to be substantially increased”. [7].

Unfortunately, this idea was followed by the impossibility of realizing it due to the limited size of the cache. But, keeping this concept of "data already saved and quick to read" and combining the structure of Pandas DataFrames, I went to create pseudo "caches" partitioned according to the information contained therein. Thanks to these pre-executed queries, we have reduced the delay in answering by removing the query interrogation of the database. Caches are related to:

- Copy numbers with relative gene name
- Families joined with the Classification table
- Names of the species considered cancer prone (cancer rate \geq 5%)
- Names of the species considered cancer resistant (cancer rate $<$ 5%)
- Names of genes that are common among cancers prone
- Names of genes that are common among cancer resistant
- All the data contained in the species table

Each of these files contains the result of the related query inside a DataFrame. These structures help us as they are already indexed and able to perform many types of different joins with structures of the same kind. Thanks to this syntax and these structures, the Full Text Search function manages at runtime what data the user is looking for, which files to read to recover this information, and how to combine them.

The first step that is performed when sending an FTS request is to break down the words contained within the primary two keywords. Thanks to the roots that we have given to the field names of the tables, the script can understand where to find the fields the user is looking for. In addition to this, objects named filter are created in which the constraints relating to the fields specified by the user will be stored. Once all the fields have been analyzed, N threads will be created where N is the number of caches, which contain the indicated fields, to be recovered. This allows the function to recover these files at the same time and to avoid a sequential reading of multiple files with a consequent dilation of the times. Once this first phase is finished, we will find in RAM the tables necessary for the construction of the answer but with a part of data not requested by the user. Consequently, the second step is to apply the filters specified for each table to reduce their size immediately. This allows the system to manipulate structures with smaller dimensions, speeding up the operations that we are going to perform on them.

The third step is to merge this data. This happens thanks to the merge function integrated with the Pandas' libraries. The type of fusion that occurs between these tables is an inner join.

If the user has also specified the exclude keyword, an additional table will be created in the same way as the previous one. The fields used to choose the caches to be recovered will be those indicated in the show, and the parameters specified in the exclude brackets will be managed in the same way as those of the filter key. Once this further table has been created, it will be subtracted from the previously generated one, creating a single result that considers all the constraints indicated by the user. For the execution of this subtraction, Pandas comes to our aid with its integrated function called concat. This allowed us to concatenate the two tables and, through a drop, to eliminate the tuples coinciding with those of the first table.

Before returning the data to the EDA model created by my colleague Fabio Bove and after the last step, the data is further manipulated. This alteration consists in the removal of the columns that have not been indicated by the keyword show, in the modification of the arrangement of the columns (the string columns will be moved to the left) to facilitate the reading of the data and, according to the column, an ordering ascending or descending of the data. We can observe a scheme of the steps previously described in the following image:

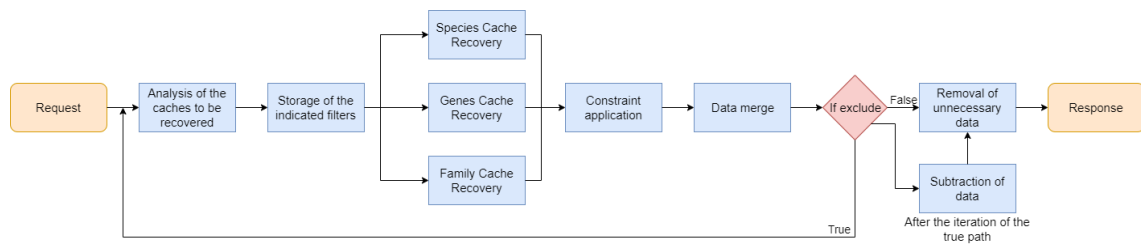


Figure 18 Calculation steps performed to return the user query result of the Full Text Search

Returning to the form relating to this part of the platform, the data are shown graphically, thanks to the EDA models seen previously. The graphs change according to the data searched by the user.

We can see these differences in the following images, taking advantage of them to give examples of queries:

- To search for the genes and the respective copy numbers of the Homo Sapiens genome, we will write:

```

show:(specie_name, gene_name, copy_numbers_qta)
filter:(specie_name=HOMO_SAPIENS)
  
```

The modal and its graph will be:

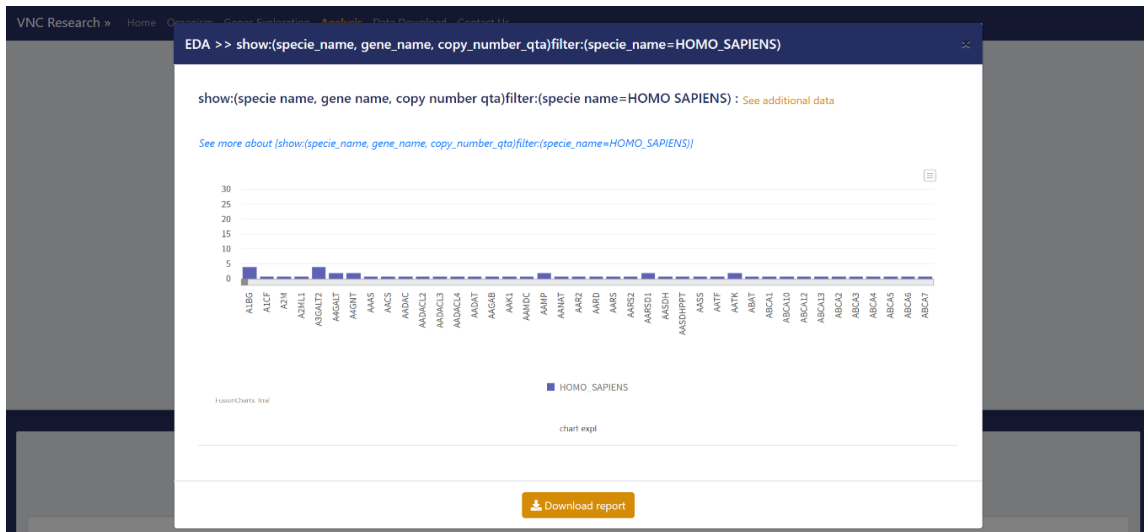


Figure 19 Result of the query described above

- To see all the species that contain the TP53 gene and its copy numbers, we will write:

```
show:(specie_name, specie_name, copy_number_qta) filter:(gene_name=TP53)
```

The modal and its graph will be:



Figure 20 Result of the query described above

The previous examples show how much, by changing a simple filter, can change the structure and the information of the result.

We can conclude that the combination of Full Text Search and EDA models has the objective of providing a statistical analysis of the data, preparing for future implementations thanks to the malleability and the data analysis tools integrated with the pandas' structures, central in the return of these data. The graphs and the information provided in this way allow researchers a quick and intuitive recovery of the data they are looking for, allowing more precise and faster research in the biological field.

3.5 Scalable Data Science

Over the years, data analysis has taken on a central role for the decisions taken by any modern company to such an extent that, to implement this technology, they do not hesitate to adopt machine learning, artificial intelligence and statistical methods within their products and in the management system of the company itself. The scale on which these analyses are applied leads to start creating new terms that can describe the phenomenon. One of them is "Scalable Data Analysis". [8].

This term indicates that data analysis applied to those data collections distributed over multiple infrastructures. This is done to support the exponential growth of the information available to us. An example very close to the case study considered by this thesis can be found in an article related to Scalable Data Analysis applied for metagenomics. It claims: *"With the advent of high-throughput DNA sequencing technology, the analysis and management of the increasing amount of biological sequence data has become a bottleneck for scientific progress. For example, MG-RAST, a metagenome annotation system serving a large scientific community worldwide, has experienced a sustained, exponential growth in data submissions for several years; and this trend is expected to continue. To address the computational challenges posed by this workload, we developed a new data analysis platform, including a data management system (Shock) for biological sequence data and a workflow management system (AWE) supporting scalable, fault-tolerant task and resource management. Shock and AWE can be used to build a scalable and reproducible data analysis infrastructure for upper-level biological data analysis services."* [9].

The aforementioned article states that a data analysis system, able to manage large quantities of data thanks to its scalability, can prove to be an extremely more powerful tool than the same local solutions. This innovation must make us reflect on the speed with which the size of the data available to users is expanding and how, consequently, the information hidden therein needs to innovate the methods used so far to extrapolate them. This new phase of data analysis is proposed as a disruptive point for future discoveries and for the existing applications, which will complement it, such as our platform.

The tangible benefits that have come back, thanks to the implementation of these strategies, are the reduction of the time required to execute and return a query.

The speed-up that we can find in the forms of the Home, Organism, and Exploration pages

was obtained, as mentioned above, by introducing the threads. This allowed us to go through the lighter forms (Home and Exploration), because of a lower amount of information returns, from an average of 7 seconds per response to an average of 2 seconds, less than a third of the time previously used. In the following image, we can see the time taken for the return of the data by the form of the Exploration page:

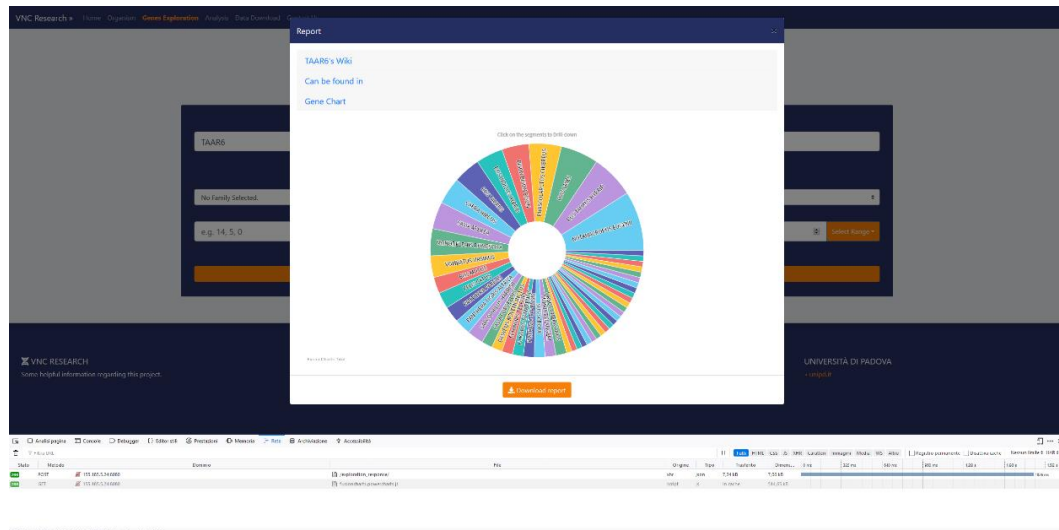


Figure 21 Time needed to return the form Exploration response recorded using Firefox's development tools

While in the following graph, we can observe more clearly the difference between the time spent from the two versions:

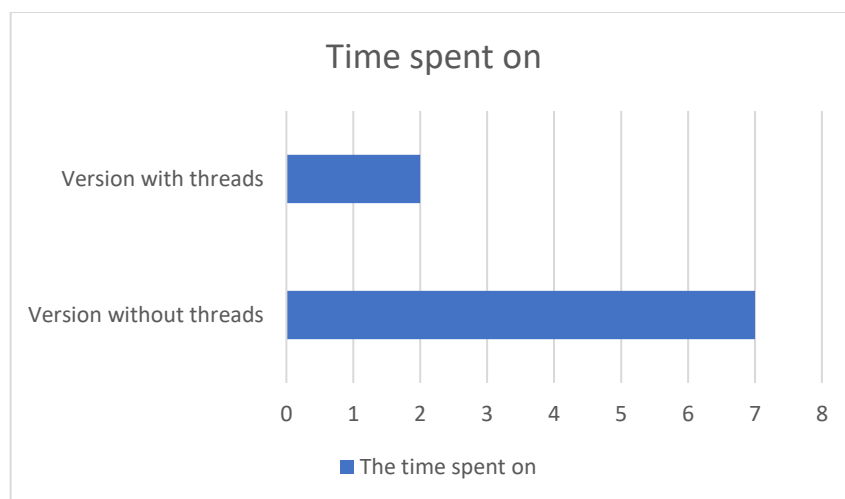


Figure 22 Difference between the times of the two versions of the form Exploration

For the form concerning the organisms, it was not possible to obtain the same instantaneousness of the response, given the 7 seconds needed after the improvement, but it was possible to get closer to it and to eliminate the previous slowness of the form that needed 24 seconds on average for a response. In the following image, we can observe the times recorded by this latest version:

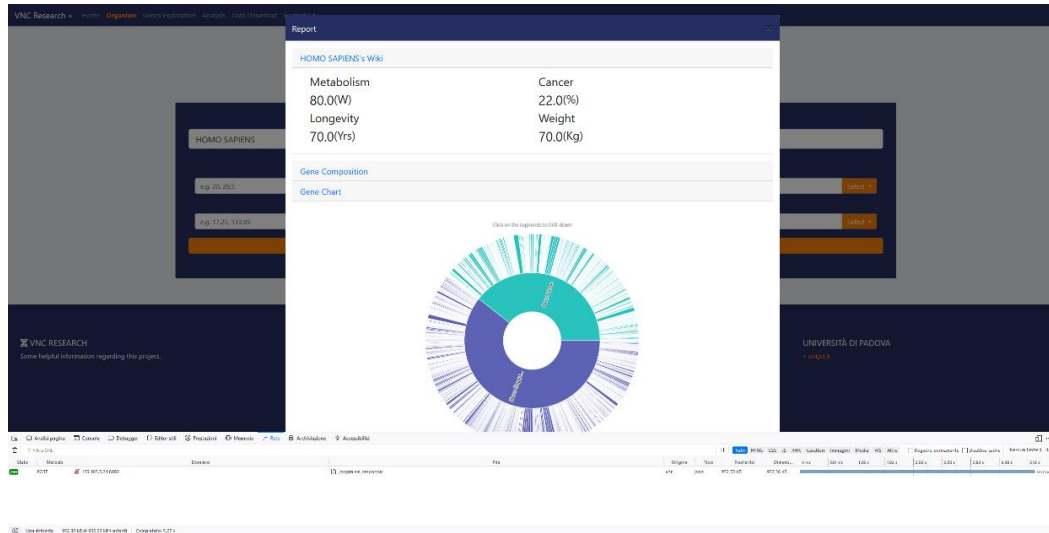


Figure 23 Time needed to return the form Organism response recorded using Firefox's development tools

Comparing the timing of the two versions considered, we can see an improvement very similar to that of the lighter forms. We note the achievement of a shorter time that is a third of the previous one:

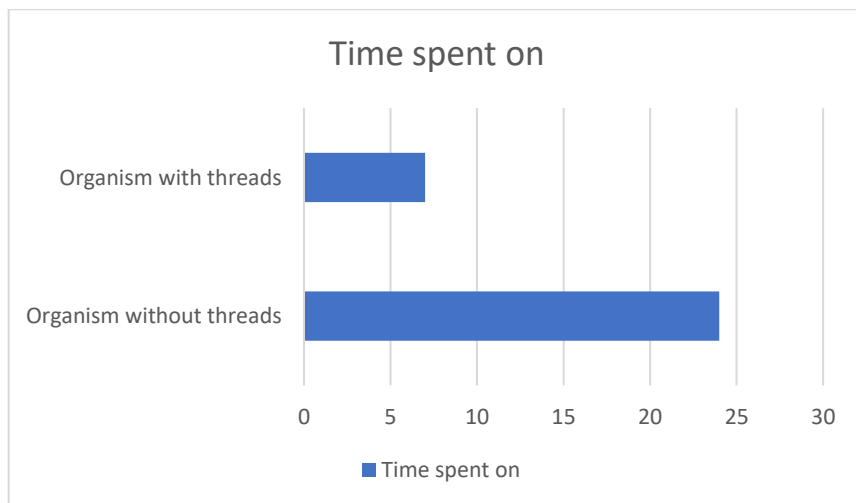


Figure 24 Difference between the times of the two versions of the form Organism

As far as Full Text Search is concerned, caches are involved in addition to threads. These bring the greatest benefit to this functionality by avoiding the query execution on the database at every request. The elimination of this step leads to a sharp reduction in the timing due to the processing of the query and to hypothetical congestion of the database due to multiple requests. The time required depends on the number of tables involved in the user query. Taking into account the worst case, the involvement of all the tables in the database, we went from an average of 25 seconds to one of 3 seconds. In the following image, we can see an example of such a query:

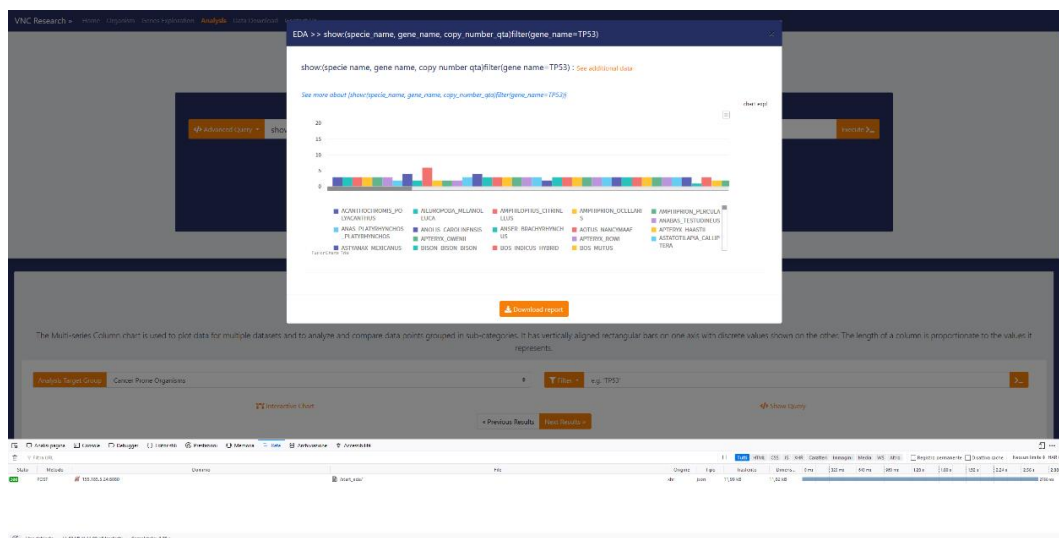


Figure 25 Time needed to return the form Analysis response recorded using Firefox's development tools

In the following chart, we can see the difference between these two versions. Compared to the other forms, there are a greater gap thanks to the further optimization made through the cache. The time now taken by the Full Text Search function is an eighth of the previously implemented version.

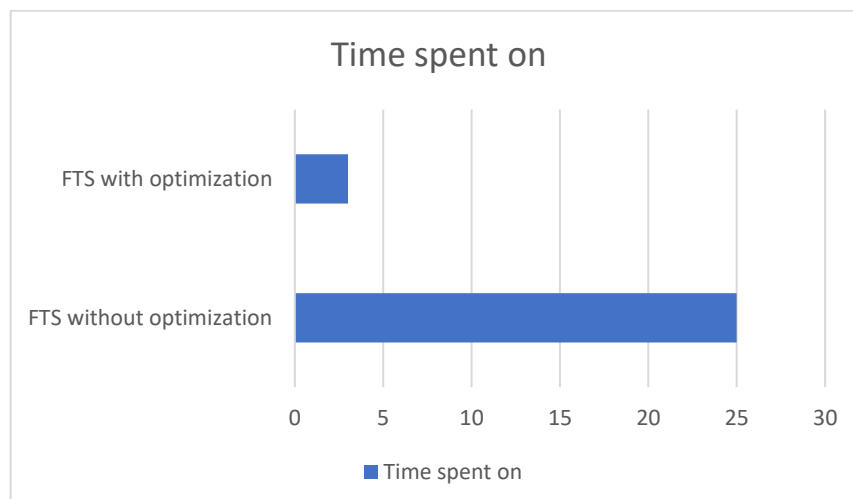


Figure 26 Difference between the times of the two versions of the form Analysis

We can see how through the introduction and the study of these new optimizations, these have led from unacceptable response times to real-time timings, omitting the connection delay, leading to a big improvement in the use of the platform.

4 Conclusions

As far as we know, there are still too many things and dynamics related to DNA and its components that we do not know yet. The thesis proposes a platform that tries to facilitate the researches in reading the relationships among the information that humanity has collected so far. It also sets itself the goal of growth together with the sector to which it is aimed, trying to become a tool on which researchers can rely.

All of this is to allow the research to continue and to give quick answers to the current mysteries about the causes of tumors onset, and the characterizing mechanisms of some species, including *Homo sapiens*.

Bibliography

- [1] A. M. B. a. C. C. M. Marc Tollis, "Peto's Paradox: how has evolution solved," *BMC Biology*, pp. 1-2, 2017.
- [2] Data Analysis - Analytics | DataSilk., "Data Analysis - Analytics | DataSilk.," 20 Marzo 2020. [Online]. Available: <https://datasilk.com/data-analysis/>.
- [3] Pandas, "Pandas," 20 Marzo 2020. [Online]. Available: <https://pandas.pydata.org/>.
- [4] D. C. Y. S.-C. L. C. S. C. Patrick Shicheng Chen, "Toward an IT investment decision support model for global enterprises," *Computer Standards & Interfaces*, pp. 1-11, 2018.
- [5] S. P. Claudio Jommi, "Public administration and R&D localisation by pharmaceutical and biotech companies: A theoretical framework and the Italian case-study," *ScienceDirect*, pp. 1-14, 2007.
- [6] N. Watkinson, "Australia – the premier Asia-Pacific hub for biotech investment," *biotech focus*, pp. 192-197, 2008.
- [7] A. J. Smith, "Cache Memories," *ACM Computing Surveys*, pp. 1-58, 1982.
- [8] A. K. a. N. T. Alon Halevy, "SCALABLE DATA SCIENCE: A NEW RESEARCH TRACK CATEGORY AT PVLDB VOL 14 / VLDB 2021," *ACM SIGMOD Blog*, p. 1, 2020.
- [9] J. W. N. D. W. G. A. W. F. M. W. Tang, "A scalable data analysis platform for metagenomics," in *IEEE International Conference on Big Data*, Silicon Valley, CA, 2013.