The Advantage of NoSQL Databases over SQL Databases

Mario-Tudor CHIRIAC
Bucharest University of Economic Studies
Faculty of Cybernetics, Statistics and Economic Informatics
Bucharest, Romania
chiriacmario21@stud.ase.ro

The aim of this article is to show the advantages of No-SQL databases compared to SQL (relational) databases. Traditionally, the concept of database is implicitly associated with relational databases, most often ignoring the huge potential that non-relational databases have in the fields such as BigData, Data Analysis, Artificial Intelligence. Besides, non-relational databases are also remarkable for their structure flexibility, their speed and for the distribution of their data on multiple servers simultaneously, for their fast writing and reading speed, which is crucial when analysing very large volumes of data.

***Keywords**: Relational databases, non-relational databases, No-SQL databases, SQL databases, SQL, No-SQL, redundancy, efficiency, scalability, portability, large volume of data, big data, management of large volumes of data, big data flow.*

1 Introduction

Before the invention of computers and electronic devices, information would be written on paper, stone, or tree bark. These methods are quite inefficient compared to the electronic storage, as they involve a high human presence in the process of writing and managing the information. Besides, the risks associated with these methods are high: damage to materials (paper, stone, wood) leading to information losses. It is almost impossible to recover the information unless there are backup copies. Duplicating these sources involves human resources and the process is very time-consuming.

An important step in the field of information storage was made with the invention of the magnetic tape in the 1960s. Such a magnetic tape would be read sequentially, which meant that in order to find the desired information, the entire magnetic tape had to be read. Compared to today's standards, this is outdated and inefficient.

The emergence of the Oracle corporation in 1977 facilitated the development of the Oracle Database relational database, which is this corporation's most well-known and highly appreciated product. At the end of the ninth decade of the last century (more precisely, in 1998), the notion of a "No-SQL" database was also postulated, a term coined by Carlo Strozzi [1]. Initially, this term designated those relational databases that would not use SQL as a data manipulation language. Later, with the emergence of the Cassandra and Voldemort databases in the early 2010s, this term became an umbrella name for all non-relational databases.

The invention of the personal electronic computer – Altair 8800 in the 1970s, would mark yet another important stage in the field of technology development and information management, while also being an important moment in the history of technology. This was an essential step, because it facilitated the emergence of several computer models in the following years, such as the Apple II and the IBM PC (Model 5150). The year 1995 marked the launch of the famous Windows 95 operating system, which became a successful brand of the Microsoft corporation. The Windows operating system has become one of the most important operating systems in the whole world, being used everywhere, from

personal computers to servers within production processes.

The exponential evolution of technology, its availability at the highest level and the fast digitalization have enhanced the population's perception of how information is used and stored, something that can be seen today.

Below, I will analyse some fundamental concepts that can be found in the above-mentioned databases, listing some of the studies that made a comparative analysis of these aspects.

The practical part gives the opportunity to understand some key concepts, useful in the presentation of the notions to be discussed, such as: scalability, performance, diversity of query languages.

2. Literature review

According to the specialised literature, we could say that No-SQL databases have several advantages over classic relational databases. No-SQL databases are designed with high scalability, which is essential nowadays, considering the high Internet access rate, as it is necessary for fast data processing. Instead of needing new servers, NoSQL databases can be scaled using the existing hardware. Thus, they can handle traffic growth and respond to demand without activity interruptions. Through this scaling, NoSQL databases become more robust and more performant, thus becoming a preferred choice for expanding data volumes. [2], [15]

In terms of performance, No-SQL databases work optimally even on devices equipped with a common hardware configuration, while relational databases require complex configurations, being specific to devices that are not readily available to the general public (Oracle Exadata) [3]. Thus, it could be said that non-relational databases are very useful for software developers and smaller businesses that do not have complex hardware resources, as they can fit on modestly-configured devices.

The data manipulation language within relational databases is quite orderly and centralised, which applies to most relational databases (Oracle Database, MySQL, MariaDB, SQLite). The differences in syntax/structure are minimal for general queries, and they occur only in advanced functionalities such as multimedia, geographical facilities, etc. No-SQL databases provide a very high flexibility. There is no well-defined standard for the language syntax, as it varies from one model to another.

One of the most well-known languages is the MongoDB Query Language (MQL) used for MongoDB document-oriented databases [4]. This decentralised approach to query languages provides several advantages for both developers and programmers. One advantage is a better cooperation between programmers and developers, thus, a continuous flow of feedback being ensured, on functionalities, bugs, and potential improvements. Besides No-SQL databases are specifically designed to be able to manage huge amounts of data efficiently, unlike the classic relational models, where their rigorous structure and integrity restrictions make such processing impossible. More detailed information about No-SQL databases, and their use in the field of big data, shall be developed in the future chapters.

According to the article [3], in the US, an SQL developer earns $84,328 per year, in average, while a No-SQL developer earns $72,174 per year. Economically speaking, the SQL developer earns more, but the No-SQL developer has the advantage of flexibility provided by the vast scope of his knowledge. A No-SQL database expert can also work as a data analyst, where the average wages are much higher than those of a simple SQL or No-SQL developer [5].

3 Theoretical concepts related to SQL and No-SQL databases. Introduction to BigData

With the invention of the computer and the Internet, the world has experienced an irreversible and revolutionary transformation of the framework in which the development of our personal lives, economic processes and activities take place. The symbiosis between the Internet and the computer has triggered the acceleration of economic and cultural globalisation. The need for rapid and instant access to information and data is the basis of today's notions and fields of activity, such as: data science, data analysis, data mining, etc.

All these sciences form the basis of the so-called BigData. It is obvious that data has a physical support on which it is stored and, via computer networks, it is transmitted between a computer and several other computers, thus forming a network.

Therefore, a database is formation of one or more interdependent data collections, accompanied by a file describing, globally, the data and the links between them. [6],[16]. Nowadays, the most common and frequently used types of databases are: SQL (Standard Query Language) and No-SQL (Not Only Standard Query Language).

There are several well-established manufacturers, on the relational (SQL) and No-SQL databases, providing database management software services. The role of database management systems is of particular importance, because, such a tool allows us to take care of the administrative sides of a database (data backup and recovery, access rights management, database performance analysis) and its security (data access authorisation and control, data encryption, protection against external attacks). According to the ranking made by the db-engines.com[7] website, the first five positions are held by the following database systems: Oracle, MySQL, Microsoft SQL Server, PostgreSQL and MongoDB.

The difference between the two types of data consists of several criteria, such as: *structural criteria*: redundancy, internal data organisation structure, query language, etc. The second criterion used to differentiate the two types of databases is related to *characteristics*: efficiency, allocated memory, speed of data processing, etc.

Based on these criteria, we can make a comparison between the two types of databases: The SQL model has a rigorous and strict structure, where redundancy is minimal and controlled. The form of organisation within an SQL-type database is represented by tables, where the following relationships can exist: *1:1 – one to one, 1:m – many to many and m:m – many to many*.

All these relationships within a relational database create the database's logical diagram. In other words, the logical diagram represents the theoretical structure of the way in which data is organised and linked together.

Besides, the logical diagram also shows the complexity of a relational database, but there are also other important concepts in the field of relational databases, such as: primary key, foreign key, integrity constraints or form of normalisation.

Unlike the SQL model, the No-SQL model is more permissive from a structural and functional point of view. No-SQL databases are divided into 4 categories: document-oriented databases (MongoDB), column-oriented databases (Apache Cassandra), graph-oriented databases (Neo4j) and key-value pair-oriented databases (Redis). No-SQL databases have a wide range of uses and applications: from the e-commerce field (electronic commerce) to the BigData field, which will be addressed in this chapter. Other domains using these databases include social media, mobile apps, gaming, and streaming services. Examples of corporations using such databases include

Netflix, Uber, and Airbnb.[8]

Netflix uses No-SQL databases to store the huge amounts of customer-generated data (viewing history, customer personal data to generate customer preferences). Uber also uses a non-relational database to store customer and driver data, trip history, trip details, and real-time location. Airbnb also uses No-SQL databases to store and manage data about properties available online, users, and a customer's accommodation history. This helps process very large amounts of data for a more complex and accurate analysis, thus helping improve the services offered on the site.

In order to illustrate more theoretical notions related to a No-SQL database, I will use MongoDB, as it is the most popular database (First place in the No-SQL database category and fifth place globally according to that ranking)[7].

Document-oriented databases have the *document* as their basic form. The document types supported by this database are: JSON (Java Script Object Notation), BSON (Binary Java Script Object Notation), XML (eXtenseble Markup Language), YAML (Yet Another Markup Language). The structure of the documents is flexible, thus allowing a greater customization of the document content. Essentially, this format is an improved version of the No-SQL key-value database model, thanks to the fact that the document has a well-defined form. Redundancy in this case is allowed and even encouraged, since, in a No-SQL database, data is stored on multiple servers simultaneously. One advantage is the fact that it provides greater flexibility in searching and retrieving data, if a server in the network cannot be accessed, due to internal or external issues. This means less disruption to data access, which is essential and necessary for today's businesses. This principle is called *scalability*.

Another important aspect, nowadays, is the speed of data writing and reading. The everyday person wants to have fast access, preferably instant, to as much data as possible simultaneously, for various fields of activity (analysis, processing). This is provided by the lack of a database diagram, the lack of normalization forms, which allows the fast development of applications with No-SQL databases. Within document databases (MongoDB), the concept on of junction does not exist. An equivalent of the junction concept is nesting an action through which a new document is inserted within the reference document. This significantly helps the processing of the document. This principle is called *efficiency*.

The variety of existing No-SQL database types has not achieved a standardisation of languages. In fact, this is an advantage, as it allows programmer communities and software development companies to compete with each other, to generate creative and unique ideas, ideas that can translate into new and innovative solutions, thus bringing progress and innovation in the field of No-SQL databases. Although the field of non-relational databases is quite new, compared to that of relational databases (which started in the late 1960s, more precisely in 1969[9]), this domain has known an impressive development. Today, many companies use a non-relational database as support, as we have seen above. This feature of non-relational databases is called *open-source*.

Non-relational databases are suitable for those who want to discover the universe of database knowledge and implicitly that of data analysis. The theoretical concepts related to No-SQL databases are fewer and much easier to understand. No-SQL database terminologies are easier to grasp, thus allowing for faster understanding and in-depth learning.

Table 1. Comparison of concepts between relational and non-relational databases

	Non-relational (No-SQL)	**Relational (SQL)**
Data storage	In tables populated with data - built based on the relational model	Storage using non-relational formats
Junctions	Yes	No
Data storage	Predefined types are respected	There are no defined forms, so it is semi-structured or unstructured.
Primary key	Yes	Yes, to which ObjectID or DocumentID can be added (MongoDB)
Diagram	Dynamic, adaptable depending on the reference model	Static
Transaction management	ACID Model Atomicity Consistency Isolation Durability	CAP Model Consistency Availability Partition Tolerance
Entity name	Table	Collection

BigData is a very important field, developed over the past years. Nowadays, this field is hugely useful, because the analyses and results provided by processing huge volumes of data help improve business processes, optimize decision-making processes, help with economic and financial forecasts, etc. Thus, the BigData field designates extremely large and complex data sets, which cannot be managed or analysed efficiently using traditional data processing tools, such as spreadsheets.[10]

The increasing access to the Internet, economic growth and the increase of living standards worldwide, have irreversibly shaped the course of humanity. Through all his interactions (social, professional, economic, etc.) the Man becomes a data provider. Between 2012 and 2013, it was estimated, according to the IBM corporation, that 2.6 exabytes of data[11] were being produced daily, which means (2.17×260 bytes). Given this huge amount of data, in order to be able to process it, there are some steps that any analyst must take in order to reach conclusive and relevant results: data collection, data processing, data systematisation and then their analysis.

Data collection involves searching for data in different sources of interest. The resulting data can be structured or unstructured. Data processing is the second stage in the set of operations that the collected data goes through. This operation involves the transformation and processing the data into a standardised and easy-to-process format. The next stage is data systematization. This process involves the removal of duplicates from the data set, the removal of outliers, which, after processing, are lost, they become corrupted, inaccessible, etc. This is an essential stage for maintaining the unity and consistency of the data, so that the result provided is as relevant and appropriate as possible. The last stage, data analysis, involves the application of one or more algorithms. Some of the algorithms used for analysis are: linear regression, logistic regression, k-clustering, k-nearest neighbours.[12] In order see how these algorithms are applied and used, the implementations will be shown place in the next chapter.

Besides, the field of BigData is the basis of AI (Artificial Intelligence). Artificial intelligence is the science aiming to make machines capable of performing tasks that would require intelligence, if they were performed by humans.[13],[17]. This

definition was provided by Marvin Minsky in 1956, during a scientific conference at Dartmouth College. BigData provides the information support on which this concept is developed. This is a field that has developed over the recent years. Four models of artificial intelligence have been discovered so far: Machine Learning (ML) which is based on the principle of autonomous learning. This is done based on the introduced data set, regardless of the field (mathematical, computer science, psychological, economic, etc.). Generative AI is the most advanced model known to date. Its characteristics are: generating images, videos, sounds or code written in different programming languages. These functionalities come as a result of a more focused and deeper learning, but also as a result of integrating more advanced languages (LLM). The other two models are: neural networks and deep learning (DL). Neural networks are reconstructions of the human brain at the algorithm level. Through this concept, an attempt is made to detail and deepen the functions of the human brain. Therefore, the starting point of these notions is represented by theoretical biology concepts. The human brain is made up of multiple neurons. A neuron is a cell specialised in receiving and transmitting information [14], [18]. Structurally, a neuron comprises: dendrites, a nucleus, and an axon. Synapses are the points of contact between two neurons, where the transmission of nerve impulses occurs. The following figure shows the structure of a neuron within the human brain:

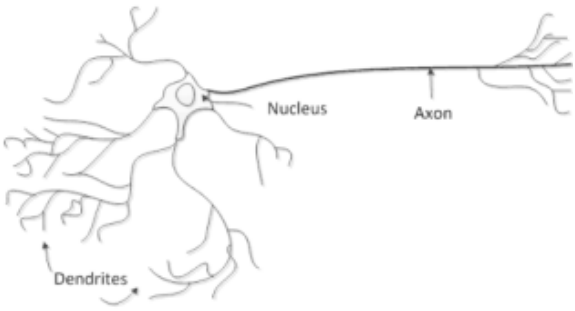

Fig. 1. The structure of a neuron in the human brain

Starting from these concepts, the first abstraction of a neural network has been created: the synapses became *inputs*, the axon became *an output*, and the nucleus became a *summator*.

Thus, the BigData field has been in a permanent and continuous evolution. Most likely, in the future we will witness even more theoretical notions and concepts that are unknown today.

4 NoSQL databases in the fields of Big Data and AI

In this chapter, we will present a case study: building a web application supported by a No-SQL database, provided by MongoDB, which will retrieve data about some customers from an XLSX file, and, data will then be recorded in the database. Based on the analyses made, the web programme will determine the customer's behaviour, thus suggesting offers for future purchases. This application can be useful for the marketing department of an e-commerce store. In order create the user interface, we will use Vue.js, and, in order to create the server connecting the database to the interface, we will use Node.js based on the JavaScript language. The advantage of web applications lays in their flexibility, as they can be run on any type of device (tablet, mobile phone, laptop, desktop, etc.), but also because most of these devices can be connected to the internet. Besides, another advantage of web applications resides in the variety of programming languages or frameworks available on the market to satisfy the needs of each programmer. Among the most famous frameworks we can mention: Vue.js, React.js, ASP.NET, Angular.js, Express.js, etc. The Vue.js framework will allow us to build a modern and user-friendly web interface. Vue.js is based on the HTML, CSS and JavaScript languages, for building objects (buttons, text writing areas, etc.) and for defining their behaviours (performing the operation shown by the button, etc.). CSS has the role of styling the web page with a modern,

pleasant and user-friendly design.

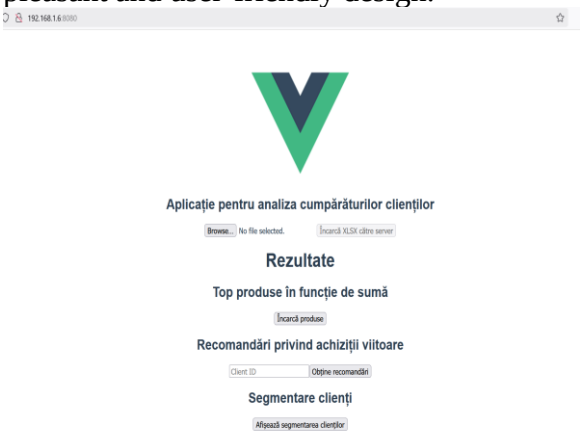

Fig. 2. Example of a user interface made with Vue.js

All of the application's operations are performed by the server (reading the XLSX file, inserting data into the MongoDB collection and the analytical data processing). In order to exemplify one of the analytical operations applied to the data set, such as ranking the most frequently purchased products, we will use, as a basis the following code fragment made in Node.js:

```
app.get("/analytics/top-products", async (req, res) => {
  let client;
  try {
    client = await connectToDb();
    const db = client.db(dbName);
    const collection = db.collection(collectionName);

    const topProducts = await collection
      .aggregate([
        { $group: { _id: "$ProductID", totalSales: { $sum: "$PurchaseAmount" } } },
        { $sort: { totalSales: -1 } },
        { $limit: 5 },
      ])
      .toArray();

    res.json({ topProducts });
  } catch (error) {
    console.error("Eroare la preluarea produselor:", error);
    res.status(500).send({ error: "Eroare de server" });
  } finally {
    if (client) await client.close();
  }
});
```

Fig. 3. Route making a connection to MongoDB, then applying processing operations, using MongoDB aggregations.

This route connects to the database, then to the desired collection. An aggregation is then applied using the .aggregate([…]) operator. Then, using the $group operator, the records are grouped according to the $ProductID field. After that, the amount spent by the buyer during the analysed period is calculated. The $sort operator sorts the data set in descending order, according to totalSales. Subsequently, the data is sent to the frontend, in JSON format, where it will be displayed. The frontend implementation will have two components: a JavaScript part represented by a function that will retrieve the data and a HTML part that will manage the display on the web page. The function retrieving the data from the server is the following:

```
async fetchTopProducts() {
      try {
        const response = await
axios.get("http://localhost:3000/analy
tics/top-products");
        this.topProducts =
response.data.topProducts;
      } catch (error) {
        console.error("Error fetching
top products:", error);
      }
    }
```

The specific HTML code for providing the final result is:

```
  <section>
        <h2>Top produse în funcție de
sumă</h2>
        <button
@click="fetchTopProducts">Încarcă
produse</button>
        <ul v-if="topProducts.length">
          <li v-for="product in
topProducts" :key="product._id">
            ID: {{ product._id }} -
Total vânzări: {{ product.totalSales
}}
          </li>
        </ul>
    </section>
```

In the result retrieval part, we used the Axios dependency. This dependency is useful when we want to write, delete, overwrite or obtain a data set. In this case, we want to retrieve the data using the axios.get() method. Then, we handle the possible errors that may occur during data retrieval. In the area where we display the result, by clicking on the *Incarca produse (Load products)* button, we will be able to see the results, because the button is marked with the @click="fetchTopProducts" event which

will refer to the JavaScript function. After that, a list of the elements obtained from the data retrieval, will be drawn up. The result will have the following form

Fig. 4. Example of result within the web application

The *k-nearest-neighbours* algorithm is one of the most well-known algorithms in the field of BigData processing and in the field of artificial intelligence. This algorithm is used in classification and regression problems. It is considered one of the easiest algorithms in the field of *machine learning*. The implementation stages of this algorithm are: choosing the number of k neighbours closest to a reference point. Then, the distance between any two points in the plane is calculated until the entire range of points is covered. The most commonly used formula to calculate the distance is the Euclidean distance. There are other types of distances as well: Manhattan distance, Minkowski distance, Hamming distance: which is used for categorical data. After calculating the distance, the shortest Ks to the reference point are chosen.

Thus, we will implement this algorithm to highlight the following scenario: generating the potential preferences of a customer in the store based on their previous purchases. Let's assume we choose k=5, that is, the number of suggestions for the customer. Therefore, we will build the function calculating the distance, with the formula for the Euclidean distance.

```
const calculateDistance = (vectorA,
vectorB) => {
  return
Math.sqrt(vectorA.reduce((sum, val, i)
=> sum + Math.pow(val - vectorB[i],
2), 0));
};
```

Now, we implement the route within the server. Due to the fact that the code is very large, I will present it in several different figures, as follows:

```
app.get("/analytics/knn-recommendations/:clientId", async (req, res) => {
  const clientId = parseInt(req.params.clientId);
  const k = parseInt(req.query.k) || 5; // Setăm implicit numărul de sugestii implicit
la 5
  let client;

  try {
    client = await connectToDb();
    const db = client.db(dbName);
    const collection = db.collection(collectionName);

    // Preluăm toți clienții cu cumpărăturile lor

    const clients = await collection.find().toArray();
    const targetClient = clients.find((c) => c.ClientID === clientId);

    if (!targetClient) {
      return res.status(404).send({ message: "Clientul nu s-a găsit." });
    }

    // Construim vectorul caracteristic unde vom salva sugestiile

    const featureVectors = clients.map((c) => ({
      clientId: c.ClientID,
      vector: [
        c.PurchaseAmount || 0,        ...(c.ProductCategory ?
[c.ProductCategory.length] : [0]),      ],
    }));

    const targetVector = [
      targetClient.PurchaseAmount || 0,
      ...(targetClient.ProductCategory ? [targetClient.ProductCategory.length] : [0]),
    ];
```

Fig. 5. Part I – connecting to the database and building the characteristic vector.

```
    // Calculăm distanțele folosindu-ne de funcția specifică a calculului distanței

    const distances = featureVectors
      .filter((fv) => fv.clientId !== clientId)
      .map((fv) => ({
        clientId: fv.clientId,
        distance: calculateDistance(targetVector, fv.vector),
      }));

    // Găsim cei mai apropiați K vecini

    const nearestNeighbors = distances
      .sort((a, b) => a.distance - b.distance)
      .slice(0, k);

    // Preluăm datele celorlați clienți

    const recommendations = [];
    for (const neighbor of nearestNeighbors) {
      const neighborPurchases = await collection.find({ ClientID: neighbor.clientId
}).toArray();
      recommendations.push(...neighborPurchases.map((p) => p.ProductID));
    }

    res.json({ recommendations: [...new Set(recommendations)].slice(0, 5) }); //
Returnăm noile sugestii

  } catch (error) {
    console.error("Eroare:", error);
    res.status(500).send({ error: "Eroare de server" });
  } finally {
    if (client) await client.close();
  }
});
```

Fig. 6. Calculating the distance and selecting the neighbours.

Now, we will see the result for a customer,

having an ID with the value of 1001.

```
{
    "recommendations": [
        "P2001",
        "P5003",
        "P9003",
        "P5007",
        "P10010"
    ]
}
```


Fig. 7. Example of result within the web application using the k nearest neighbours algorithm

Of course, these algorithms can also be used in other fields of study (medicine, education, networking, etc.), or, in our case, for other scenarios: optimising results based on preferences/Internet searches/budget allocated by the consumer. This way, the online advertising provided by merchants will be more relevant to the consumer. In order to have the expected, desired results, we must have as much data as possible, which also has to be relevant. The premises that we want to analyse must be logical and they must have a well-defined purpose, so that the decision-maker can always make the best decision.

5 Conclusions

In conclusion, No-SQL databases have greatly contributed to deepening and and consolidating the BigData field. With their flexible and scalable way of managing data, non-relational databases prove that they can be an alternative to traditional relational databases, which have a more rigid structure and which are not so flexible, when it comes to processing large volumes of data. MongoDB databases also stand out in BigData processing, as they provide a new perspective on the process of storing and analysing data. The use of data collections and nesting within documents allows for easy and fast data management, compared to the traditional relational models, which do not benefit from so much flexibility and contain more theoretical concepts, as seen in the previous chapters. Besides, the use of operators in data processing is an advantage, as they are intuitive and easy to use, being of great help to new programmers in this field. In addition, the study of No-SQL databases is an important step for any programmer, entering a new path of knowledge. Besides, the syntax of the MongoDB operators, is also easy and intuitive. Machine-learning algorithms along with the processing of large volumes of data make up a symbiosis that will develop the study and research in any scientific field, providing as many solutions as possible, to questions that have not yet been answered.

References

[1] https://www.telacad.ro/cursuri/baze-de-date-nosql-redis-mongodb-neo4j/

[2] https://www.oracle.com/ro/database/nosql/what-is-nosql/, accessed on 21.03.2025

[3] https://www.guru99.com/ro/sql-vs-nosql.html, accessed on 21.03.2025

[4] https://ro.simeononsecurity.com/articles/sql-vs-nosql-choosing-the-right-database-management-system/, accessed on 21.03.2025

[5] https://www.unite.ai/ro/ce-este-un-inginer-de-date-foaie-de-parcurs-responsabilit%C4%83%C8%9Bi-salariale/, accessed on 21.03.2025

[6] coord. I. Lungu, Adela Bâră, Constanța Bodea, Iuliana Botha, Vlad Diaconița, Alexandra Florea și Anda Velicanu, *Tratat de baze de date vol. I: Baze de date, organizare, proiectare, implementare*, Editura ASE, București, 2011, p. 77

[7] https://db-engines.com/en/ranking, accessed on 12.12.2024

[8] https://www.datastax.com/guides/nosql-use-cases, accessed on 12.12.2024

[9] coord. I. Lungu, Adela Bâră, Constanța Bodea, Iuliana Botha, Vlad Diaconița, Alexandra Florea și Anda Velicanu, *Tratat de baze de date vol. I: Baze de date, organizare, proiectare,*

implementare, Editura ASE, București, 2011, p. 77
[10] https://www.oracle.com/ro/big-data/what-is-big-data/, accessed on 15.12.2024
[11] https://web.archive.org/web/20130824213031/http://www.ibm.com/big-data/us/en/, accessed on 16.12.2024
[12] https://tdwi.org/articles/2018/07/02/adv-all-5-algorithms-for-big-data.aspx, accessed on 17.12.2024
[13] https://www.sap.com/romania/products/artificial-intelligence/what-is-artificial-intelligence.html, accessed on 17.12.2024
[14] https://code-it.ro/introducere-in-retele-neuronale-teorie-si-aplicatii/, accessed on 18.12.2024
[15] https://codegym.cc/ro/quests/lectures/ro.questhibernate.level19.lecture00, accessed on 21.03.2025
[16] https://support.microsoft.com/ro-ro/topic/no%C8%9Biuni-elementare-despre-bazele-de-date-a849ac16-07c7-4a31-9948-3c8c94a7c204, accessed on 21.03.2025
[17] https://www.europarl.europa.eu/topics/ro/article/20200827STO85804/ce-este-inteligenta-artificiala-si-cum-este-utilizata, accessed on 21.03.2025
[18] https://askabiologist.asu.edu/neuronii-%C5%9Fi-nervii, accessed on 21.03.2025

CHIRIAC Mario-Tudor First year student, in the Master's Programme named: "Databases - business support", within the Faculty of Cybernetics, Statistics and Economic Informatics - the Bucharest University of Economic Studies. In 2024, I graduated from the Bachelor's programme - Economic Informatics in Romanian, within the Faculty of Cybernetics, Statistics and Economic Informatics at the Bucharest University of Economic Studies. Areas of interest: study of SQL and No-SQL databases, programming web applications using Vue.js, React.js.