A Semantic NoSQL Application Program Interface for Big Data

K. ElDahshan[1], E. K. Elsayed[2], H. Mancy[3], A.AbuBakr[4]
Department of Mathematics-Faculty of Science, Al-Azhar University, Cairo, Egypt[1]
Department of Mathematics-Faculty of Science (Girls), Al-Azhar University, Cairo, Egypt[2,3,4]
Computer Science Institute, Canadian International College, Cairo, Egypt[2]

***Abstract*—Complexity, heterogeneity, schemaless-ness, data visualization, and extraction of consistent knowledge from Big Data are the biggest challenges in NoSQL databases. This paper presents a general semantic NoSQL Application Program Interface that integrates and converts NoSQL databases to semantic representation. The generated knowledge base is suitable for visualization and knowledge extraction from different Big Data sources. The authors use a case study of the COVID-19 pandemic prediction and other weather occurrences in various parts of the world to illustrate the suggested API. The Authors find a correlation between COVID-19 spread and deteriorating weather. According to the experimental findings, the API's performance is enough for heterogeneous Big Data.**

***Keywords*—NoSQL database; formatting; semantic technology; data integration; pandemic prediction**

I. Introduction

Large data sets have their roots in the 1960s and 1970s when the world of data was just getting started with the creation of the first data centers and the development of the relational database. Nowadays, the data grows at sky-high rates. The world has multiple Big Data sources with different structures like sensors, scientific experiments, and social networks. We produce and collect more data every minute, and we need to be able to process them as soon as possible. Everything depends on timing, including stock trading, tracking the growth of epidemics, and traffic monitoring. A minor misunderstanding could lead to both financial loss and fatalities. The high level of adoption of Big Data technology is influenced by the quick and continual growth in data volumes.

A. Research Problem

Researchers face many problems when dealing with data from multiple sources. Because the structure of the data is different it makes it difficult to process the data and extract knowledge. Most researchers are now trying to store the data in a semantic form to make it easier to process. To do this they need to collect data from different data stores and preprocess these data to present them in semantic form. The preprocessing phase takes much time and effort which we need to solve the real problems we face and take better decisions based on the knowledge extracted from the data.

This research is funded by the Academy of Scientific Research and Technology (ASRT), Cairo, Egypt, project titled “Coronavirus Prevalence Prediction Model” (Project ID: 6641).

B. Research Objectives

This article studies many semantic Big Data frameworks and their limitations. Then it proposes a semantic NoSQL application program interface (API). The proposed API can read data from multiple NoSQL databases and convert them to Ontology. By using the proposed API, it will be possible to apply semantic queries on different data stores. The proposed API can also be used to integrate different data stores into a defined format.

C. Organization

This paper is organized in the following way to fulfill the research goals: Section 2 reviews a background about Big Data, Ontology, and NoSQL databases. Section 3 examines the history of earlier similar works. Section 4 Shows the proposed API architecture. The implementation of our proposed API is described in Section 5. Section 6 provides a case study of the API to analyze COVID-19 spread and weather behavior. The impact of utilizing the suggested API is described in Section 7. The paper's conclusion is covered in Section 8, which also analyses the research's significant contribution and limitations.

II. Background

A. Big Data

The term Big Data refers to a concept mostly used to classify large amounts of data. Despite the wide agreement on the promises and prospects of Big Data, there is no standard definition for it at this time.

The Institute of McKinsey Global defined the term Big Data as a “very Big Data set that can’t be stored or managed with database software tools” [1].

Gartner [2] defined Big Data as "a high-volume, high-variety, and/or high-velocity information asset that necessitates cost-effective, novel types of data processing that provide better insight, decision-making, and process automation."

The term "Big Data" was defined by Deepak Gupta and Rinkle Rani [3] as ‘Big data refers to large datasets which require non-traditional scalable solutions for data gathering, storage, management, analysis, and visualization, to extract actionable insights that could have an impact on every area of human life’. Big data characteristics are portrayed as extensions of ’V’s [3][4]. Fig. 1 presents the Big Data characteristics.

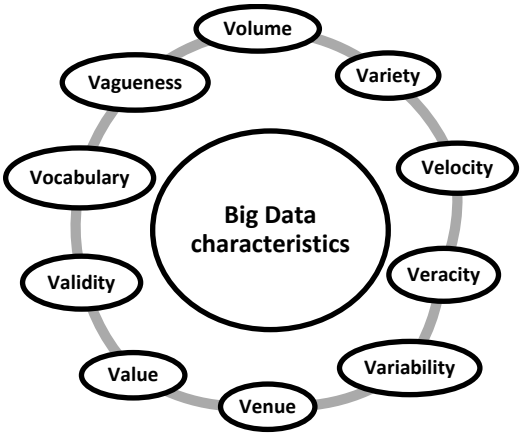

Fig. 1. Big Data characteristics

B. Ontology

An Ontology consists of O = A, C, I, P, R, T. Where A is the set of axioms; C is the set of concepts or classes; I is the set of instances; P is the set of properties of the concepts describing various features; R is the set of relationships between concepts and T is the set of hierarchical relationships among concepts that are called taxonomies [5].

The components of Ontology are as follows:[6]

- *Individuals*: "The ontology population is represented by individuals, which are objects. They are instances of classes."
- *Classes*: "Classes are particular categories of objects or things that represent a collection of instances or specify a particular category of entities. They are frequently referred to as concepts or entity types. Classes can be used to classify individuals, other classes, or a combination of both."
- *Attributes*: "Properties associated with objects or classes are called attributes. They include statements about Datatypes and their DataValues, characteristics, features, or parameters of individuals and classes."
- *Relations*:" The numerous links that indicate how two individuals or classes are related. Additionally called associations, roles, relationship types, and object properties".
- *Axioms*:" Logical rules and assertions that together create the general theory that describes the relationship between the ontology elements. They represent formal definitions of the ontology knowledge."

C. NoSQL Databases

NoSQL (Not Only SQL) is a database that is a non-relational distributed database system. It facilitates the rapid structuring of data analysis with large volumes of data and a variety of data types. NoSQL is also referred to as a cloud database, a non-relational database. NoSQL databases are not based on tables and do not often employ structured query language to manipulate data. Receive and append operations are frequently highly optimized in NoSQL database systems. When working with large amounts of data and this data structure does not require a relational model, NoSQL databases come in handy [4].

When compared to relational databases, NoSQL databases are more scalable, diverse, simple to use, flexible, and give better performance. MongoDB is now the most popular NoSQL database, with Apache Cassandra, Redis, and HBase following closely behind. Neo4j is the most popular NoSQL graph database and the most common cloud database is Amazon DynamoDB [7]. NoSQL data models allow related data to be stored in a nested data structure [8].

Based on their data model, NoSQL databases are classified into a variety of types[5]. The main types of NoSQL databases are wide-column databases, document databases, key-value databases, and graph databases. They provide flexible schemas and they are very scalable to large amounts of data and high user loads. Table I provides a comparative study of the four types of NoSQL databases.

TABLE I. A COMPARATIVE STUDY OF THE FOUR TYPES OF NOSQL DATABASES

NoSQL database types	Document database	Key-value store	Wide-column store	Graph store
Data Storage	It stores data in JSON, BSON, or XML documents	It stores data as an attribute name (or "key") combined with its value	It stores data in tables, rows, and dynamic columns	It stores data in nodes and edges
Use Cases	It is great for a wide variety of use cases and can be used as a general-purpose database	It is the best to use when you need to store large amounts of data but you don't need to perform complex queries to retrieve it such as storing user preferences or caching	Common use cases for wide-column stores include storing Internet of Things data and user profile data	It is the best to use when you need to traverse relationships to look for patterns such as fraud detection, social networks, and recommendation engines
Performance and scalability	High	High	High	Very high
Flexibility	High	High	Moderate	High
Complexity	Low	Very Low	Low	High
Examples	MongoDB, CouchDB	Redis , DynamoDB , Voldemort	Hbase, Big Table, Cassandra	Neo4j, HyperGraph, InfiniteGraph

III. Related Work

Many researchers studied the relationship between semantic technology and Big Data technology. They tried to connect knowledge management systems with NoSQL database management systems to apply semantic queries on Big Data.

Bansal S and Kagemann S proposed Semantic Extract-Transform-Load (ETL) framework that uses semantic technologies to integrate and publish data from multiple sources. The Extract-Transform-Load (ETL) process refers to a process in data warehousing that extracts data from outside sources, transforms it to fit operational needs, which can include quality checks, and loads it into the end target database. The authors extracted data from different sources in flat file formats such as CSV. The proposed semantic ETL framework first creates a semantic model of the datasets being integrated, and then it creates semantically linked data that adheres to the data model. A semantic data model and semantically linked data (RDF triples) are produced using semantic technologies and stored in a data warehouse during the transform phase of an ETL process. The transformation phase will involve a manual process of analyzing the datasets, the schema, and their purpose. Based on the findings, the schema will have to be mapped to an existing domain-specific Ontology or Ontology will have to be created from scratch [9].

Hanen Abbes and Faiez Gargouri implemented a tool to generate Ontology from MongoDB databases. It proposed transformation rules from MongoDB to OWL Ontology. This work is done in five main steps. First is the creation of the Ontology skeleton by defining Ontology classes and detecting the relationships between them. Second, learn object properties and datatype properties. Third, identify Individuals. Forth, deduce class axioms, property axioms, and constraints. Finally, enrich the Ontology with class definition operators [10].

Mahmudul Hassan and Srividya K. Bansal proposed a solution to execute SPARQL query as SQL query using Apache Spark on large-scale RDF data stored in NoSQL databases such as HBase and Cassandra. It translated the SPARQL query to SPARK SQL for both HBase and Cassandra storage schemas. It first converted RDF data to store it in HBase and Cassandra and then proposed an algorithm to convert SPARQL query to SPARK SQL using the in-memory data processing engine Spark. The main purpose of this paper is to execute a query on RDF data (semantic data) with a large volume. To achieve that, it stored the RDF data in a NoSQL database. However, the algorithm can be used to perform SPARQL queries on data already stored in HBase or Cassandra [11].

K. ElDahshan, E. K. Elsayed, and H. Mancy developed a semantic dashboard using java JDK to connect to HBase and built a universal knowledge base using Protégé. It converted the SPARQL query to spark SQL using Sempala. The conversion is done using the algebra tree [12].

S. Mhammedi, H. El Massari, and N. Gherabi proposed an approach to automatically learn OWL Ontology from data in the Couchbase database by applying six mapping rules. The mapping rules are learning classes, learning object properties from the embedded document, learning datatype properties, transforming all data values of fields in each document to individuals, learning property restrictions, and learning class hierarchies [13].

All of them except [11,12] do not handle Big Data. All of them allow the conversion of only one NoSQL database type. The proposed API handles Big Data, accepts any type of NoSQL database, allows schema conversion, and the proposed API is platform-independent. Table II shows the comparison among related works through data sources, NoSQL type, tools used, and limitations.

TABLE II. A Comparative Study of the Related Works

Research	Data Source	NoSQL Type	Tools	Limitations
[9]	Data from multiple sources in flat file formats such as CSV	--	Semantic Extract-Transform-Load (ETL) framework which generates a semantic model of the datasets under integration and then generates semantically linked data	The framework can not deal with NoSQL database stores
[10]	MongoDB	Document database	A tool to generate Ontology from MongoDB implemented by the JAVA programming language	The tool can deal with only one type of NoSQL database store
[11]	HBase and Cassandra	Wide-column	A query compiler that is written in Flex and Bison that translates SPARQL to Spark SQL to execute a query on RDF data stored in HBase and Cassandra	The tool can deal with semantic data in RDF format stored in a wide-column database
[12]	HBase	Wide-column	A semantic dashboard using java JDK to connect to HBase and built a universal knowledge base using Protégé	The tool can deal with only one type of NoSQL database stores
[13]	Couchbase	Document database	An approach to learning Ontology from the Couchbase database using mapping rules. This work is done by the JAVA programming language and OWL API	The tool can deal with only one type of NoSQL database stores

IV. The Proposed Semantic Integration Application Program Interface Architecture

Researchers and academics regard data access to be unattainable in many situations for a variety of reasons. These include the following: a profusion of data, non-computerization of processes, heterogeneity, data duplication, and the presence of a lot of isolated data in databases that can only be accessed in a specific context. These traits typically lead to low-quality information, which makes it challenging for researchers to organize and evaluate them during the decision-making process [14]. Data integration is the process of making it possible for people to access, deliver, and utilize data from several sources and huge businesses while preserving its integrity and quality. Real-time updates to data saved in one source can also be mirrored in other sources thanks to this [15].

Although Ontology development is not a new field of study, Big Data faces new difficulties due to its features (velocity, variety, and volume). Therefore, our API takes into account the properties of Big Data for Big Data Ontology creation. The main idea is to create Ontology from a large number of diverse sources. The authors' goal is to make the process of automatically generating Ontology and importing Big Data as simple as possible for the user by providing an independent application program interface. The research's additional significant contributions include (1) the ability to integrate different sources of NoSQL, (2) the ability to support the Ontology creation-based Big Data access layer, (3) the ability to re-engineer and combine different Ontologies, and (4) the capability to easily build a semantic query over Big Data. Fig. 2 shows the proposed API overview.

The structure of the proposed API is as follows:

- *Connection Phase*: In this phase, the authors build a Big Data access layer that allows users to connect to any NoSQL database.
- Integration and Extraction *Phase*: In this phase, the API reads and extracts every record in the database and puts them in a Python dictionary.
- *Conversion phase*: In this phase, the API converts the dictionary to an XML file.
- *OWL Creation phase*: In this phase, the API creates OWL from the XML file.
- *Semantic and visualization phase*: In this phase, the API builds semantic rules and visualizes the data in a graph.

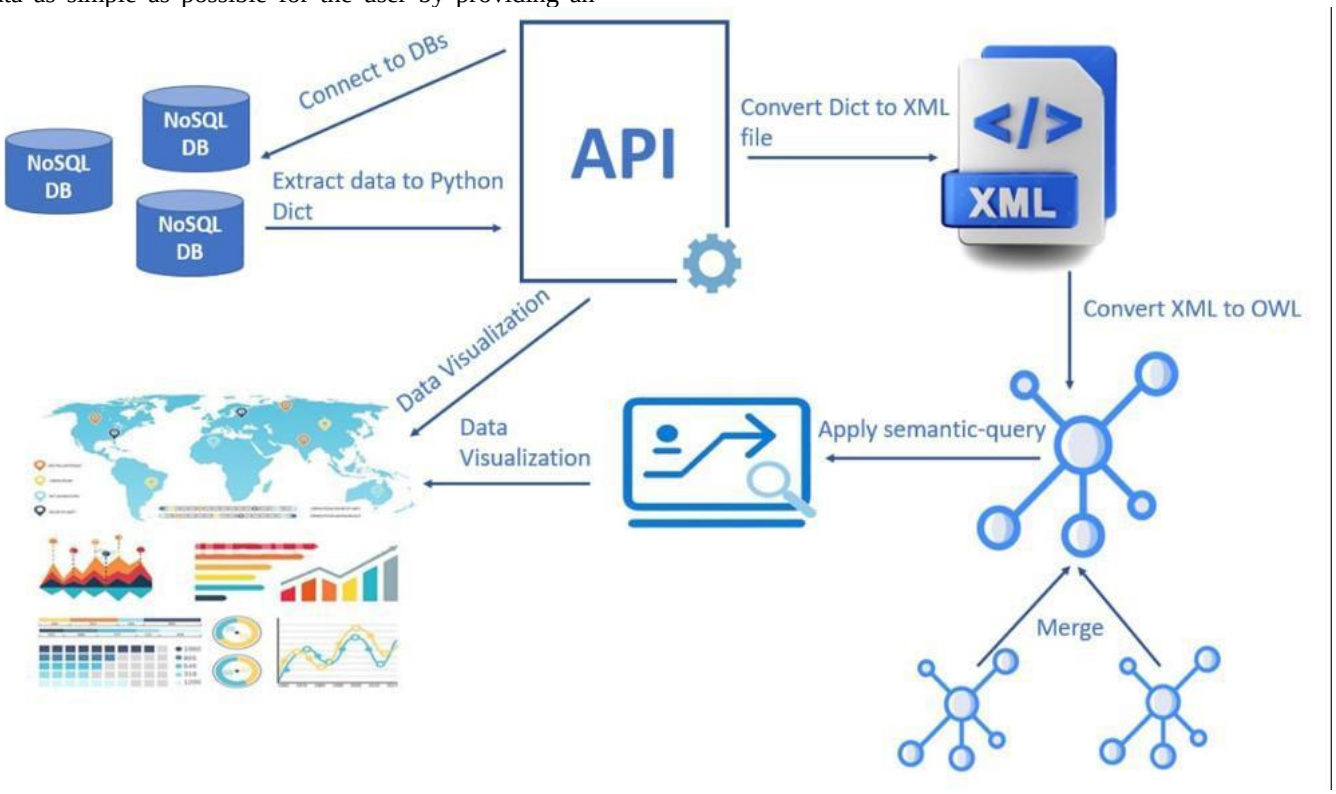

Fig. 2. The proposed API Overview

VI. Proposed Semantic Integration Application Program Interface Implementation

The API is created with Python 3.8.5 on Windows 10 operating system with Intel(R) Core(TM) i7-6500U CPU and 8GB RAM. The proposed API asks the user to enter the databases' names to be converted. The API scans the databases and retrieves every record in them with the same structure as it was in the database. After getting every record, it will be converted to XML. The API can handle data with any structure. Every record will be retrieved as a dictionary data type in Python. Then it will be converted to a JSON array. The JSON array will be converted to XML using the 'dicttoxml' package. The API can handle documents with complex structures and large sizes. For every collection in MongoDB and every table in HBase, a root class will be created. A subclass will be created to represent each attribute in the records. If the document in the MongoDB collection has embedded objects, other subclasses will be created to represent the attributes of the embedded object. After the conversion is completed, the XML is converted to OWL Ontology using DTD2OWL. Fig. 3 explains the proposed API workflow. Algorithm 1 demonstrates the Semantic NoSQL API execution steps.

Algorithm 1: Semantic NoSQL API

```
Initialize
Connect to the database
Scan every record in the database
For (every record) do
    Put the record in Python dictionary
    Convert the dictionary to XML
    Save the output to a file
    End
Convert XML to OWL
Build semantic Query
Visualize the result
```

The phases of the proposed Onto-NoSQL API are implemented as follows:

A. Connection Phase

To access a NoSQL database, the author used the pymongo package to connect to MongoDB as a document database type, the happybase package to connect to HBase as a Wide-Column database type, the Redis-py package to connect to Redis as a Key-value database, and the Neo4j Python Driver to connect to Neo4j as a Graph database type. Algorithm 2 represents the connection code to MongoDB. Algorithm 3 represents the connection code to Hbase.

This paper focuses on both document database stores and column-oriented database stores as they are the most used NoSQL database stores. The most popular document store is MongoDB [6]. MongoDB is very popular because it allows multiple data types to be used. The database consists of collections. The collection stores the data as JSON documents. Schema is not needed to store the data in MongoDB. The data stored can be a string, number, date, array, or object. MongoDB can store data from multiple sources in different formats.

Algorithm 2: Connect to MongoDB

```
from pymongo import MongoClient
client = MongoClient()
#enter database name
database= input('Enter DB name:')
db = client[database]
```

One of the most popular column-oriented database stores is HBase. HBase is an open-source management system that is a versioned and distributed database based on Google's BigTable.

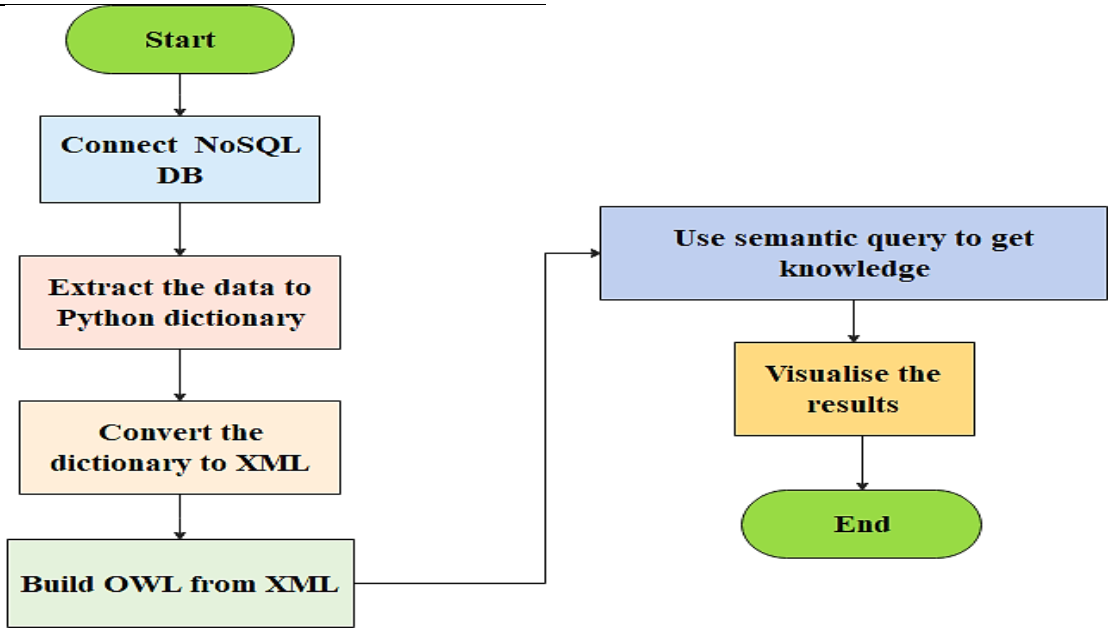

Fig. 3. The proposed API workflow

This system is column-oriented and built on top of HDFS, which speeds up read and write operations across Big Data sets. Application programming interfaces (APIs) such as Thrift and Java provide access to HBase. There are no query or scripting languages specific to these APIs. HBase is reliant on a ZooKeeper instance by default [16].

Algorithm 3: Connect to HBase

```
import happybase as hb
conn= hb.Connection(HostName,PortNumber)
conn.open()
```

B. Integration and Extraction Phase

The proposed API integrates MongoDB and HBase databases. It extracts every record in the database and puts them in a Python dictionary. Algorithm 4 reads records from MongoDB. Algorithm 5 reads records from HBase.

Algorithm 4: Read records from MongoDB

```
collection_names = db.collection_names()
For (every collection ) do
    #retrieve documents in each collection
    docs = db[collection_name].find()
    End
```

Algorithm 5: Read records from HBase

```
table = conn.table('table_name')
For (every key, row) do
    # retrieve records
    row = str(row)
    row =json.loads(row)
    End
```

C. Conversion Phase

The proposed API converts the dictionary to an XML file. Algorithm 6 converts the dictionary generated from MongoDB records to an XML file. Algorithm 7 converts the dictionary generated from HBase records to an XML file.

Algorithm 6: Convert dictionary generated from MongoDB to XML file

```
#create XML file
f = open(database+'/'+collection_name+'.xml', 'wb')
For (every doc) do
    #convert object id to string
    doc_sanitized = json.loads(json_util.dumps(doc))
    #convert to XML
    xml = dicttoxml.dicttoxml(doc_sanitized)
    f.write(xml)
    End
f.close()
```

Algorithm 7: Convert dictionary generated from HBase to XML file

```
#create XML file
f = open('table_name.xml', 'wb')
For (every key, row) do
    xml = dicttoxml(row)
    f.write(xml)
    End
f.close()
```

The interface of the API was created using PHP (Laravel Framework). First, we need to install the Process component that executes commands in sub-processes. This can be done using the composer to install the package. Second, use the Process class that enables Laravel to run a script. Third, create a new instance of the Process class, which takes three parameters the name of the script, the script itself, and the arguments passed to the script. An HTML form was created to read the data from the user. The user chooses the NoSQL database engine (MongoDB or HBase) and writes the name of the database (in the case of choosing MongoDB) or the table (in the case of choosing HBase) to be converted to XML. After the script is executed, the user gets a message that the database (in the case of using MongoDB) or the table (in the case of using HBase) is converted successfully. The XML files are located in the public folder in the Laravel project. Fig. 4, 5, and 6 represent the screens of the proposed API. Fig. 7 is a snapshot from the generated XML file.

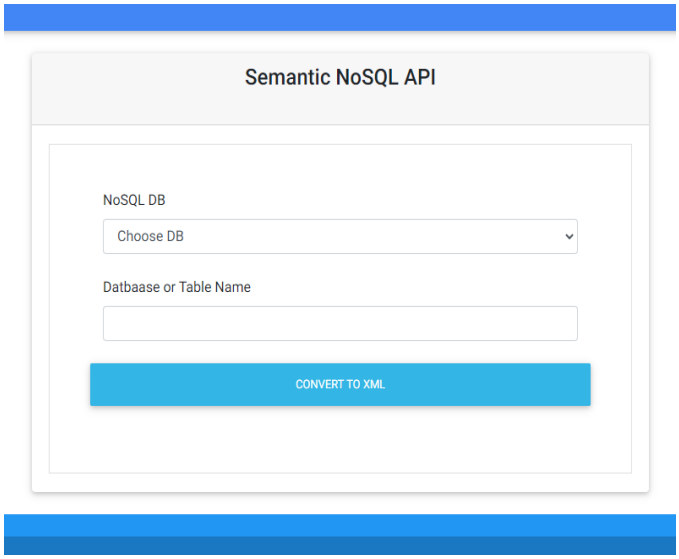

Fig. 4. The API screens- the input form

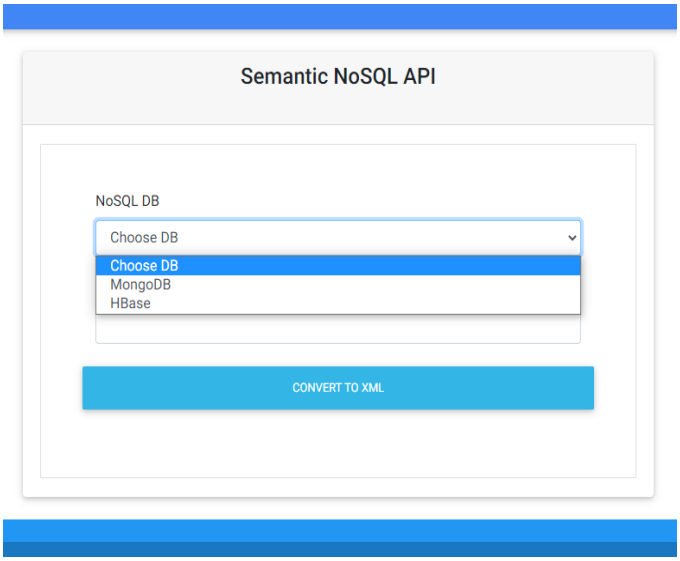

Fig. 5. The API screens- choose NoSQL data store

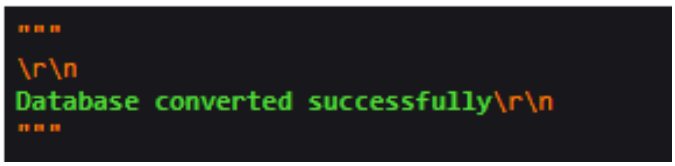

Fig. 6. The API screens- the success message after the database is converted

<?xml version="1.0" encoding="UTF-8"
?><root><doc><_id type="dict"><key name="$oid"
type="str">61c481b0593caffbcd8229d9</key></_id><Co
nfirmed type="int">0</Confirmed><Country
type="str">Afghanistan</Country><Date
type="str">2020-01-22</Date><Deaths
type="int">0</Deaths><Recovered
type="int">0</Recovered></doc>
<doc><_id type="dict"><key name="$oid"
type="str">61c481b0593caffbcd8229da</key></_id><Co
nfirmed type="int">0</Confirmed><Country
type="str">Afghanistan</Country><Date
type="str">2020-01-25</Date><Deaths
type="int">0</Deaths><Recovered
type="int">0</Recovered></doc>
<doc><_id type="dict"><key name="$oid"
type="str">61c481b0593caffbcd822c93</key></_id><Co
nfirmed type="int">157795</Confirmed><Country
type="str">Afghanistan</Country><Date
type="str">2021-12-20</Date><Deaths
type="int">7335</Deaths><Recovered
type="int">0</Recovered></doc></root>

Fig. 7. A snapshot of the generated XML file

D. OWL Creation Phase

The proposed API converts XML format to OWL. The Authors implement the DTD2OWL2 Method in python [17].

E. Semantic and Visualization Phase

The proposed API builds a semantic query using rdflib [18] and visualization using the Plotly package in Python.

VII. Case Study

The authors apply API to COVID-19 and weather data. Data was collected from various sources regarding the spread of Covid19. The number of cases was collected from the WHO. The weather data was collected from National Centers for Environmental Information website [19]. The authors used the number of confirmed cases, population, and weather data to predict a new pattern of confirmed cases and to find a relation between it and the other factors. The data was preprocessed and converted to JSON format. MongoDB was used to store the confirmed case data. HBase was used to store the weather data. By converting the database to OWL, it is possible to apply semantic queries and extract more knowledge from the data. It was possible to visualize the data and find a relation between them despite being stored in different databases with different structures. Fig. 8, 9, and 10 visualize the data on the world map.

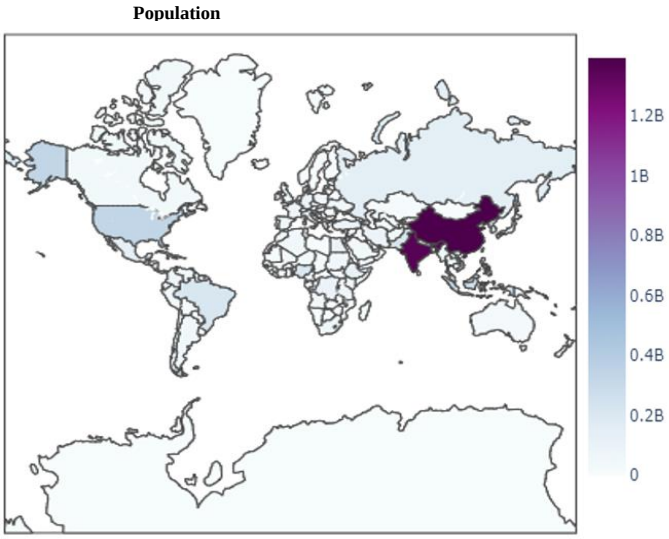

Fig. 8. The population of world countries

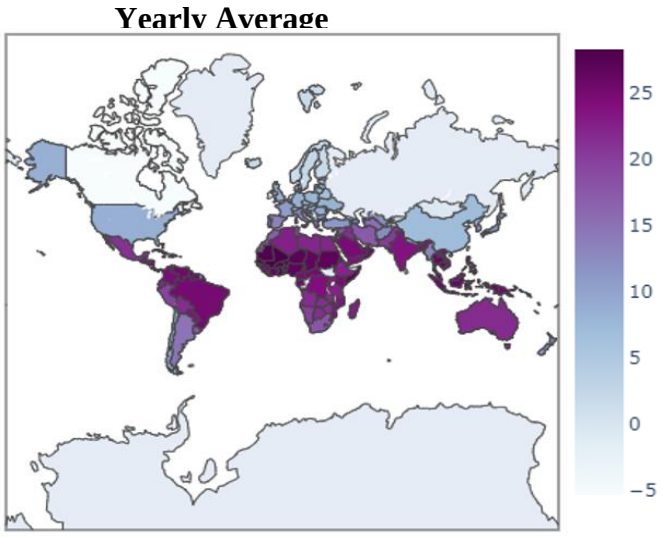

Fig. 9. The yearly average temperature of the world

The Covid-19 data was used to implement the proposed API. These data are various, change rapidly, and have large sizes. The data type can be text describing the case or symptoms, numbers describing the number of infected cases, images of the X-ray performed on the patients, or geographical data describing the spread of the virus. All these data can be stored in NoSQL databases. The user can work with one type of stored data to predict the behavior of the virus or even diagnose a patient based on the X-ray performed on his chest. The user needs to integrate these data from multiple sources to achieve his goal. The data can be integrated to extract more knowledge from them and help countries to prepare and deal with the spread of the virus. NoSQL databases can help to improve dealing with such important and various data to make better decisions. By converting the NoSQL database to Ontology, the user can discover new rules and relationships.

VIII. Analysis of Results

The used data were in the form of text and integers that represent the number of confirmed cases of Covid19 around the world. They also represent other factors like each country's temperature and population. The interval of the collected data was over 500 days from 22nd Jan 2020 to 31st Aug 2021. The confirmed case data and the population of every country in the world were stored in MongoDB, whereas the weather data were stored in HBase. By studying Fig. 10, it may be possible to see the spread of the virus around the world. By studying other factors, it could be possible to find a pattern or even a factor that deeply affects the infection rate, which will help us find a way to decrease the infection rate.

In Fig. 11 you could see the relation between the temperature represented by the blue curve and the number of confirmed cases represented by the red curve. The data were scaled in the range of 0 to 1 using the min-max scaler

As stated before, these data were stored in different NoSQL databases with different types. By using the proposed API, it was possible to connect to these databases and retrieve data from them. Four countries from around the globe were chosen to study the effect of temperature on the spread rate of the virus. The X-axis represents the number of days while the Y-axis represents the number of confirmed cases and temperature degrees every day.

By choosing a small interval of time such as in Fig. 12, it could be concluded that the infection rate is inversely proportional to the temperature degree. Of course, other factors affect the infection rate such as the population of the country under study (Fig. 8 demonstrates the population of world countries), the educational level of most of the population that affects their behavior, or the economy of the country that affects the medical care system.

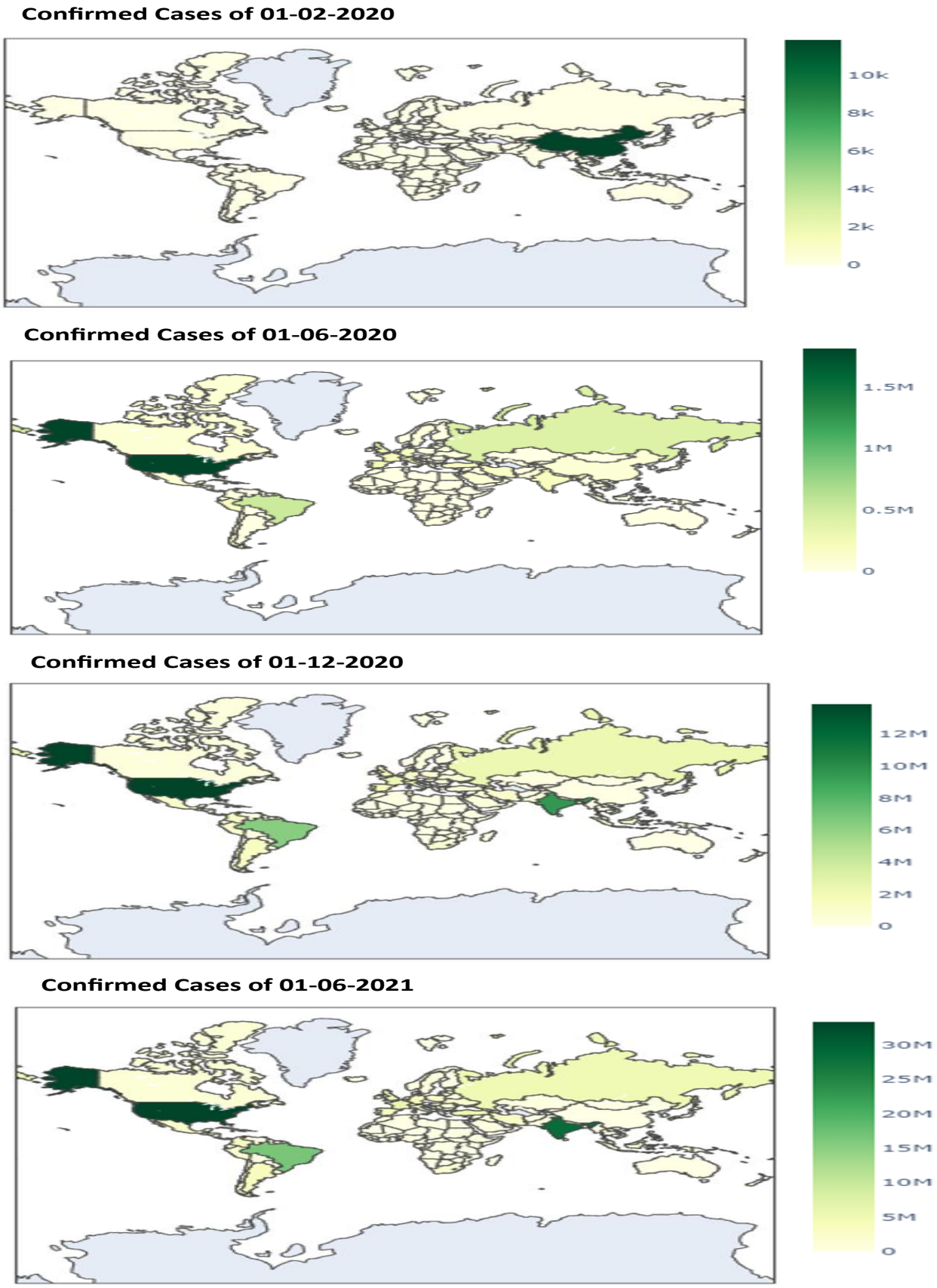

Fig. 10. The spread of confirmed cases over the world

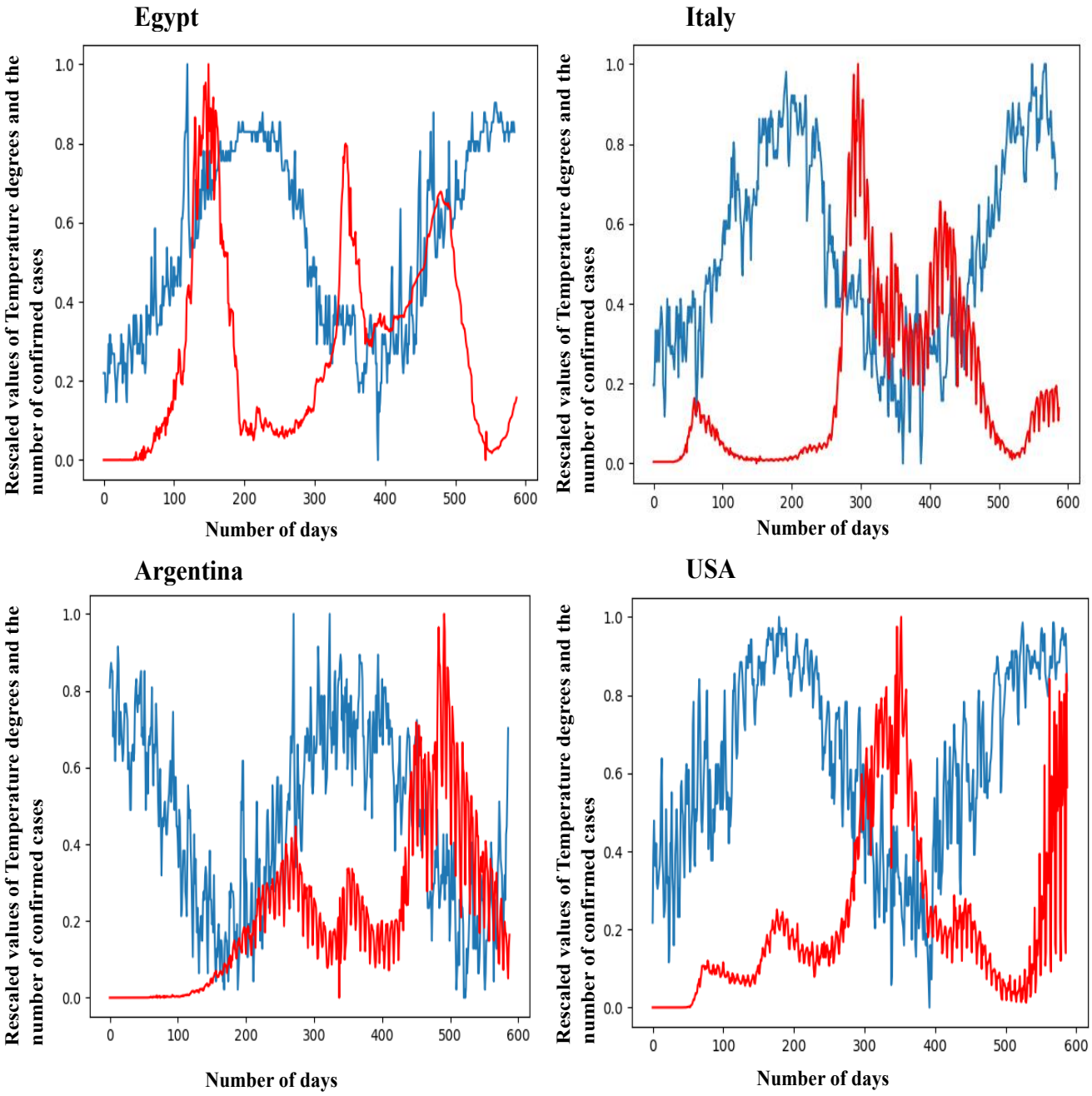

Fig. 11. The relation between temperature (blue) and number of confirmed cases (red) (The X-axis represents the number of days)

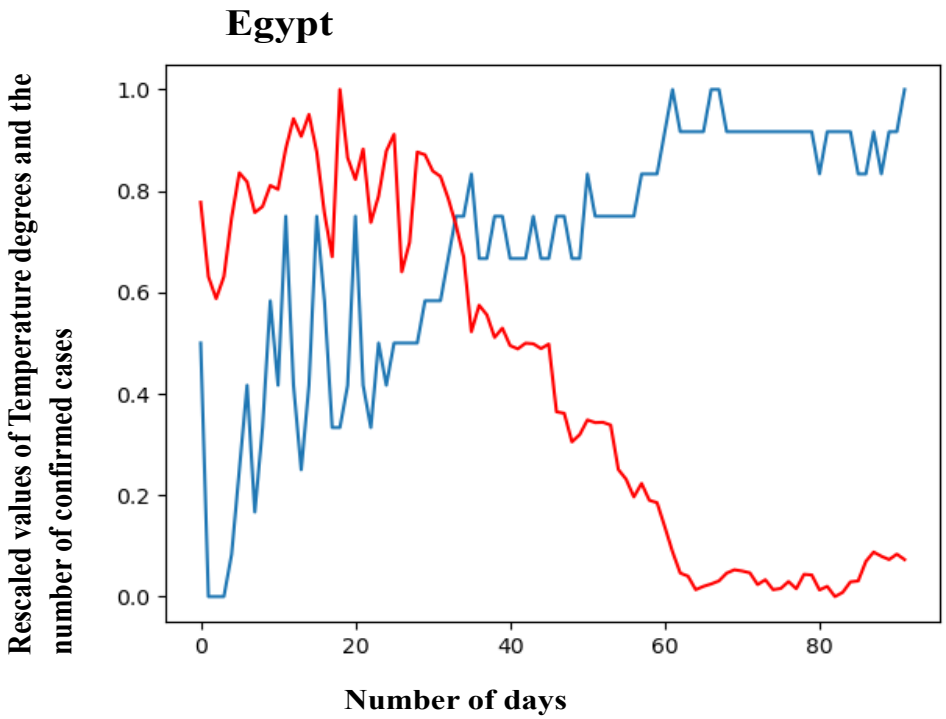

Fig. 12. The confirmed cases in Egypt during June, July, and August 2020 are in red, and the temperature is in blue (The X-axis represents the number of days)

IX. Conclusion

This paper presented Big Data concepts, technologies, and challenges. It studied NoSQL databases, their types, and the relationship between NoSQL Databases and Ontology. A proposed API was developed to integrate multiple NoSQL databases and generate OWL Ontology from them. By using the proposed API, it is possible to apply semantic queries, find new rules, and extract more knowledge. Also, it can be used to integrate data from different data stores into a unified defined format. Semantic data visualization has shown the relation between different data and discovered new patterns. One can say that the temperature is a factor in the virus's spread. It spreads faster in lower temperatures like in the USA, Russia, and some European countries (Fig. 9 demonstrates the yearly average temperature of the world). It also spreads faster in countries with a high population like India.

References

[1] Manyika J, Chui M, Brown B, et al. *Big data: the next frontier for innovation, competition, and productivity*. McKinsey Global Institute Report, May 2011.

[2] Gartner. *Gartner IT Glossary: Big Data definition*. Available from: http://www.gartner.com/it-glossary/big-data [Accessed 22-2-2021].

[3] Gupta D, Rani R. A study of big data evolution and research challenges, *Journal of Information Science*. 2019; 45(3) :322–340.

[4] Moorthy J, Lahiri R, Biswas N, Sanyal D, Ranjan J, Nanath K, et al. Big Data: Prospects and Challenges. *VIKALPA The Journal for Decision Makers*. 2015;40(1):74-96 Available from: DOI:10.1177/0256090915575450 .

[5] ElDahshan K, Elsayed E.K, and Mancy H. Enhancement Semantic Prediction Big Data Method for COVID-19: Onto-NoSQL. *IAENG International Journal of Computer Science*.2020;47(4):613-622

[6] Khadir A, Aliane H, Guessoum A. Ontology learning: Grand tour and challenges. *Computer Science Review*. 2021;39(100339). Available from: https://doi.org/10.1016/j.cosrev.2020.100339.

[7] DB engines, Available from: https://db-engines.com/en/ranking [Accessed 8-12-2021].

[8] MongoDB, Inc. *What is NoSQL?*. Available from: https://www.mongodb.com/nosql-explained [Accessed 22-2-2021].

[9] Bansal S, Kagemann S. Semantic Extract-Transform-Load framework for Big Data Integration. *Computer*. 2015 Mar 1;48(3):42-50. https://doi.org/10.1109/MC.2015.76.

[10] Abbes H, Gargouri F. Big Data Integration: a MongoDB Database and Modular Ontologies based Approach. *Procedia Computer Science*. 2016; 96:446-455. Available from: https://doi.org/10.1016/j.procs.2016.08.099

[11] Hassan M, Bansal S. Semantic Data Querying over NoSQL Databases with Apache Spark. In: 2018 IEEE International Conference on Information Reuse and Integration (IRI), Salt Lake City, UT, USA. IEEE; 2018. P.364-371, doi:10.1109/IRI.2018.00061

[12] ElDahshan K, Elsayed E Mancy H. Semantic Smart World Framework. *Applied Computational Intelligence and Soft Computing*. 2020, Article ID 8081578. Available from: https://doi.org/10.1155/2020/8081578

[13] Mhammedi, S., El Massari, H., Gherabi, N. Cb2Onto: OWL Ontology Learning Approach from Couchbase. In: Gherabi, N., Kacprzyk, J. (eds) *Intelligent Systems in Big Data, Semantic Web and Machine Learning*. Advances in Intelligent Systems and Computing, vol 1344. Springer, Cham. Available from: https://doi.org/10.1007/978-3-030-72588-4_7

[14] Lima V, Bernardi F, Domingues M, Kritski A, Rijo R P, Alves D. A computational infrastructure for semantic data integration towards a patient-centered database for Tuberculosis care. *Procedia Computer Science*. 2022;196:434-438. Available from: DOI:10.1016/j.procs.2021.12.033

[15] Ahmed J, Ahmed M. Semantic Web Approach of Integrating Big Data. *International Journal of Computer Sciences and Engineering*. Sep 2018;6(9):529-532. Available from: DOI:10.26438/ijcse/v6i9.529532

[16] Khan N, Yaqoob I, et al. Big Data: Survey, Technologies, Opportunities, and Challenges *The Scientific World Journal*. 2014, Article ID 712826. Available from: http://dx.doi.org/10.1155/2014/712826

[17] Hacherouf M, Bahloul S.N, Cruz C. Transforming XML documents to OWL ontologies: A survey. *Journal of Information Science*.2015; 41(2):242-259.Available from: DOI:10.1177/0165551514565972.

[18] *rdflib documentation*. Available from: https://rdflib.readthedocs.io/en/stable/ [Accessed 15-3-2022]

[19] National Centers for Environmental Information. Available from: https://www.ncei.noaa.gov/ [Accessed 20-12-2021]