2021 2nd International Conference on Robotics, Electrical and Signal Processing Techniques (ICREST) | 978-0-7381-3042-2/20/$31.00 ©2021 IEEE | DOI: 10.1109/ICREST51555.2021.9331044

Performance Comparison for Data Retrieval from NoSQL and SQL Databases: A Case Study for COVID-19 Genome Sequence Dataset

Soarov Chakraborty, Shourav Paul, K. M. Azharul Hasan
Department of Computer Science and Engineering
Khulna University of Engineering & Technology
Khulna-9203, Bangladesh
soarovchakraborty@gmail.com, paulshourav124000@gmail.com, az@cse.kuet.ac.bd

***Abstract*—NoSQL database management system is introduced to tackle different sorts of challenges, including performing operations on unstructured, semi-structured, and structured data. NoSQL databases gained popularity because of the improved performance than the SQL databases. We aim to investigate the NoSQL system's performance, namely MongoDB and Cassandra and SQL database, namely MySQL for DNA sequences data from the COVID-19 dataset. Studies of the DNA sequences are essential for medical diagnosis and biotechnology. However, it is quite challenging to store these genomics data in a traditional RDMS because of their unstructured nature. NoSQL is an efficient solution for textual characters like genomics data. We used around 3GB of human genome data from the COVID-19 dataset provided by NCBI. The original data was in the FASTA format, and we process these data into JSON format. Also, we have analyzed the different query syntax, data load time, and query performance time for the genomics data.**

***Index Terms*—NoSQL, MySQL, Document Database, Big Data, COVID-19 Genome Data, DNA Sequence Data.**

I. Introduction

The storage systems of traditional DataBase Management Systems (DBMS) like MySQL are less efficient in storing unstructured and semi-structured data since they can not maintain all the properties of Big data, namely value, volatility, validity, veracity, variety, velocity, volume [1] [2]. Tech giants like Facebook, Google, Amazon, and LinkedIn have their own distributed storage system to store their vast amounts of data. For storing, processing, or working with the big data, we need a more dynamic distribution storage system [3]. Therefore, to efficiently store and manage the Bigdata, distributed database systems named Not Only SQL (NoSQL) are becoming popular. For storing tremendous quantities of data, organizations namely Amazon's Dynamo [4], Google's Bigtable [5], Facebook's Cassandra [1], Ebay's MongoDB [6] have created different types of NoSQL databases for their varieties of need.

DNA is the molecular name of a compound that holds genetic guidance among all living beings. The DNA molecule consists of a double helix structure, which looks like a loop called polynucleotides. Each nucleotide contains one of the four different nitrogenous bases, i.e., thymine (T), cytosine (C), adenine (A), guanine (G). This DNA combines with the proteins and eventually transform into chromosomes. There are 23 pairs of chromosomes in the human genome. Currently discovered more than 12 types of human genome sequences, and most of them use high-performance techniques and short-read [7]–[9]. DNA contains a zettabyte of information, and to store such vast information, we need a distributed storage that can be accessed efficiently.

In this paper, we made a comparison for data load and retrieval for traditional DBMS MySQL [10] and NoSQL database system MongoDB [6] and Cassandra [1] for the massive amount of human genomics data from the COVID-19 dataset. We compared data load time and query time of the nucleotide sequences and different JSON format for the different database systems. We found the NoSQL system works well comparing to SQL DBMSes, and Cassandra shows better performance than MongoDB.

The rest of the paper is organized as follows; In section II, some related works of recent years have been explained. Section III illustrates the genomic domain for the dataset. A brief comparison of NoSQL and traditional databases are described in section IV. The comparison result is analyzed in section V. Finally, section VI outlines conclusion and future work.

II. Related works

Various methods have attempted so far for storing huge amounts of data into the NoSQL databases. Karimi et al. [11] proposed a scalable technique that can match large amounts of DNA signatures data at a rapid rate using various DBMS like Hbase, Hive, Hadoop, and NoSQL DBMS. They retrieved bacterial genomes from the NCBI dataset. Vela et al. [12] introduced a system for storing transport routes implementing crowdsourcing techniques [13] utilizing a NoSQL graph-oriented database. Their system can store those data of routes that will be generated by a mobile application. For storage purposes, their arrangement uses a NoSQL graph-oriented database. Moreover, they have addressed many different physical design tasks for durability, protection, availability, and scalability for balancing the data in their work.

Connor et al. [14] proposed a cloud-based Query Engine that retrieves information from thousands of genomes using scalable NoSQL, namely Hbase. Using different NoSQL models, Banane et al. [15] analyzed performance on the dataset of the resource description framework. For the NoSQL system, adaptive monitoring based on a cloud-enabled framework was proposed in [16] by presenting the different NoSQL models elasticity features. A comparison of MongoDB and SQL databases, including select, update, and aggregation operation, is presented in [17] for different schema and tables for both the databases and found that MongoDB performs well in every criterion except for the aggregation operation. Yishan Li and Sathiamoorthy Manoharan [18] compared different operations on SQL and NoSQL databases based on *key-value* pair framework. They showed that SQL performs better than NoSQL databases, especially in the case of a small dataset and structured data. Using the Yahoo cloud serving benchmark tool compared and analyzed data models of different NoSQL databases, namely Cassandra, Couchbase, HBase, Redis, and MongoDB [19].

In this work, we compare and analyze the human genome sequence data along with the nucleotide sequence and additionally convert the FASTA data into different JSON formats. Besides, analyzing the data load time and also queries of the nucleotide sequences for MongoDB, Cassandra, and MySQL.

III. Analysis Of Genomic Area

Study of the genome implies analyzing the evolution, function, content, and structure of the genome. Genomics means processing and storing vast amounts of data for genomes. Knowledge of the DNA sequence has become essential in several domains like medical analysis, biological research, biotechnology, biological systematics. There are 20,000-25,000 human protein-coding genomes. This creates a massive number of DNA sequences. The repetition of the sequence makes it unmanageable and creates many challenges with the genomic data. Half of the human genome is covered with repetition. That means the same pattern reappeared over and over. Moreover, there are many missing parts in the genomic data. Furthermore, this data is enormous in volume and also is partial and noisy [20]. Long DNA molecules are divided into the smaller part using many techniques, and these parts use to define nucleotides order in the DNA.

Fig. 1 explains the diagram of the genomic study. This diagram begins with the biological issues where all sorts of questions can occur. After that, bioinformatics research performed on these questions to produce the data and arrange the biological experiments. Moreover, analyze these data and store them into a database. Later the whole diagram restarts again and again with new biological issues by applying the previously stored data.

IV. Data Modeling

Data modeling is the process of planning a data model for storing and manipulating the data using a database. Scalability is a property of a system that defines its skill to operate under an extended workload. SQL and NoSQL have different scalable properties. Most SQL databases are vertically scalable. So expanding components like CPU, RAM, or SSD can increase the performance on a single server. On the other hand, NoSQL databases are horizontally scalable. They can control increased traffic by attaching more servers to the database. NoSQL databases can become more extensive and powerful, making them the preferred option for managing massive data sets. Elasticity is the feature to which a system can adapt workload changes in an on-demand fashion and balance the number of resources allocated to a service. We process the data using three steps: data acquisition, data preprocessing, and data manipulation.

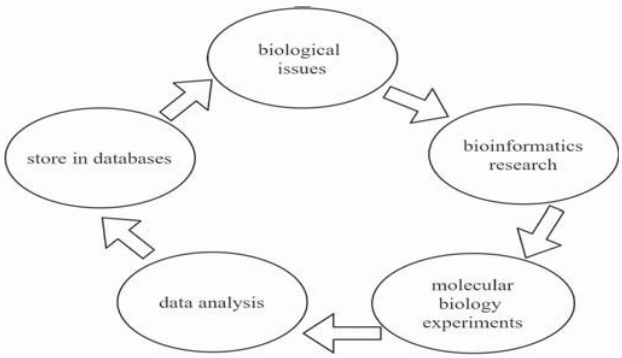

Fig. 1. Genomics analysis workflow.

A. Data Acquisition

The human genome data from the COVID-19 dataset is collected from NCBI[1]. The size is around 3GB. In the 1st phase of this dataset, nucleotides data are arranged 70 characters per line, and for the 2nd phase, these data are ordered in 130 characters per line. The original human genome data from the COVID-19 dataset are organized in the FASTA format. A sample of the dataset is shown in Listing 1. The FASTA format is a text-based format containing either nucleotide sequences or peptide sequences in which single-letter codes are used to describe base pairs or amino acids. An order in FASTA format starts with a single-line, accompanied by a series of sequence data[2].

```
>FJ376619.2 Bulbul coronavirus HKU11-934,
complete genome
GACAAAGCTCAAAATCAATACGTTATACGTATTGTATTTTGTAGCCCTC...
GACACCAATCCAGGTGCGTTGTGCTGGACAGTGCCTGCCCGGTTTAGTG...
...
TGTGAGGATGATGGTGCCGATGAAGGAGTCATACCACAGTGTTGGGATG...
TTACAAAAATTAAAAAGGATGAACCCTGCGTTGAAGTACAACAAGGTG...
...
```

Listing 1. COVID-19 genome data representation in the FASTA format.

B. Data Preprocessing

There are some missing parts in the dataset. Moreover, some portions of the data are faulty as well as noisy. Therefore,

[1]https://www.ncbi.nlm.nih.gov/datasets/coronavirus/genomes
[2]https://zhanglab.ccmb.med.umich.edu/FASTA/

machines cannot interpret them. So we had to fill up those missing values. Since the data is in FASTA format, it can not be used directly in the SQL or NoSQL models. Therefore, we organized the data so that there are more than 100,000 regions in the dataset, and each region contained at least 100 nucleotides each. We preprocessed the data into JSON format to use in MongoDB, Cassandra, and MySQL since JSON uses human-readable text to store and transmit data objects consisting of attribute-value pairs.

C. Data Manipulation

The data model defines the logical structure for a database and illustrates how data is connected and processed. We represent a brief description of the data model for MySQL, MongoDB, and Cassandra.

1) MySQL: We have used PhpMyadmin in XAMPP, which provides a GUI tool called MySQL [10]. For MySQL data modeling, we designed two tables namely *Genome* and *Sequence* using the dataset. In the *Genome* table, region_number is the primary key, and the genome_name field fills with "HumanGenome" string. For the *Sequence* table, Sequence_id is the primary key, and region_number applies as a foreign key. Also, there are other fields like Id and Nucleotides. The Id field contains a unique value for an individual region, and the Nucleotides field holds a series of characters representing a nucleotide. Fig. 2 demonstrate the schema diagram for MySQL.

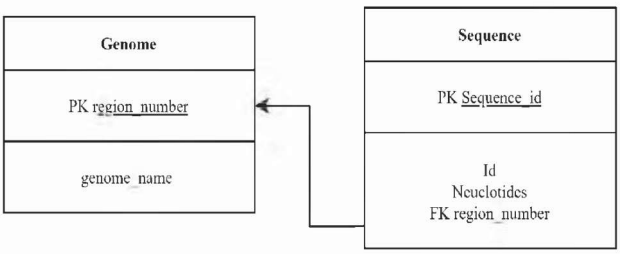

Fig. 2. ER diagram for storing the human genome.

```
{
        "_id":
        {
          "Object_Id": "5f021297decf274268c51d21"
        },
        "genome_name": "HumanGenome"
        ,
        "region_number": 1,
        "sequence":
        [{
          "id": 3,
          "nucleotides": "GACAAAGCTCAAAAT..."
        },{
          "id": 4,
          "nucleotides": "GACACCAATCCAGGT..."
        },{
          "id": 5,
          "nucleotides": "GACACCAATCCAGGG..."
        },
        ...
        ]
}
```

Listing 2. Genomics data organized in MongoDB

2) MongoDB: In the case of MongoDB, it was installed on a single server and data stored as JSON format. MongoDb supports embedded fields. Instead of an external table, this extensive list of human genome data was stored in a single document. An example of these embedded properties of MongoDB and JSON format of the stored dataset is demonstrated in Listing 2. These data are being fetched using MongoDB Compass and MongoChef. Here Object_Id field is automatically created by MongoDB. The sequence field holds the embedded property of MongoDB. Here the id field carries a different value for a unique region, and the nucleotides field holds a string that describes a nucleotide sequence.

3) Cassandra: In the case of Cassandra, we have created a table named *human_genome* for storing the human genome data from COVID-19 dataset. DataStax Bulk Loader was used for loading the data into the Cassandra table. We utilized a nested frozen map in a set and map used to store id and nucleotides as a *key-value* pair. For mapping, both the data type is text, and frozen was introduced cause it serializes all the elements into a single value. Also, Cassandra sees the frozen type values as a blob. After that, this frozen map was stored into a set and set collected these unique values in sorted order. Listing 3 demonstrates all the syntax for creating a Cassandra table, and the human genome data stored in the Cassandra table has displayed in Listing 4.

```
CREATE TABLE human_genome( id text, genome_name text,
region_number int, sequence set< frozen< map
< text,text >>>, PRIMARY KEY ( (id), region_number) )
WITH CLUSTERING ORDER BY (region_number DESC) AND
caching = {'keys':'ALL', 'rows_per_partition':'10'};
```

Listing 3. Creating human_genom table in Cassandra

```
id   | genome_name | region_number|   sequence
-----+-------------+--------------+--------------
1683 | HumanGenome |     1683     | {{'id': '1'},
{'nucleotides':'GCTCAGTTTAC...'},{'id': '2'},...}
...
```

Listing 4. Genomics data organized in a Cassandra table

V. Experimental Results

The experimental results are explained in this section. We developed prototype systems for all the three systems and executed in a machine having Windows 10 operating system with 8 GB RAM CPU and NVIDIA GeForce GPU.

A. Data Load Time

We have compared data load time for MySQL, MongoDB, and Cassandra. The data load time was calculated by 25%, 50%, 75%, and 100% of the total data, and the loading time is taken to load a particular portion of data, as shown in Fig. 3. Cassandra performs better than other DBMS, and MySQL has taken much time and performs worse among these three.

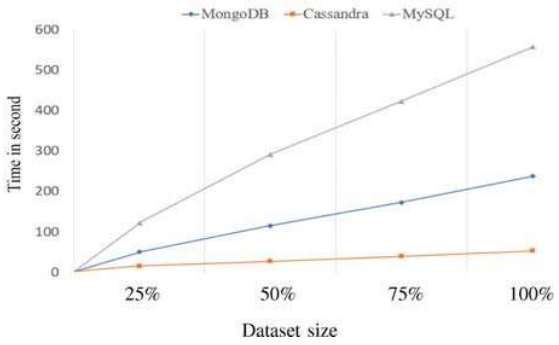

Fig. 3. Comparison of data load time for different DBMS

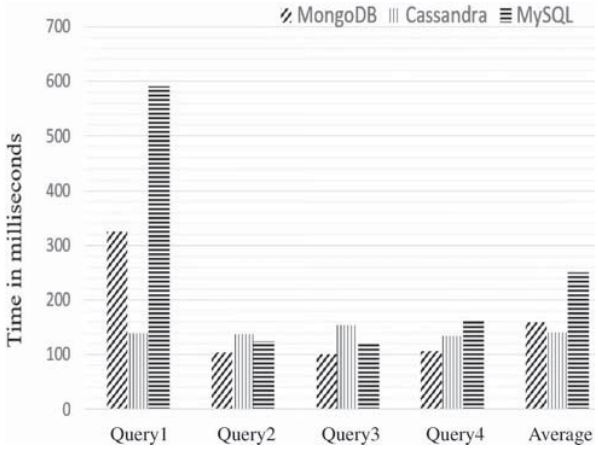

Fig. 4. Query response time (in milliseconds)

B. Query Time

MySQL, MongoDB, and Cassandra have their query syntax. Three standard queries are performed using these DBMSes. In every database, at the time of performing a query, we calculate the total time needed for that operation. In the case of MySQL, we have to join two tables, namely Genome and Sequence, for retrieving name and nucleotides using region number and sequence id. Listing 5 represents this MySQL query.

```
SELECT Genome.genome_name, Sequence.neuclotides FROM
Genome INNER JOIN Sequence ON Genome.region_number =
Sequence.region_number WHERE Genome.region_number =
4 AND Sequence.Id = 3
```

Listing 5. Query performed in MySQL

In MongoDB, we have retrieved all the information using name, region number and sequence id from the human_genome table. Listing 6 represents this MongoDB query.

```
db.human_genome.find({ $and: [{"genome_name":
"HumanGenome"}, {"region_number": 4},
{"sequence.id": 3 }]}).pretty()
```

Listing 6. Query performed in MongoDB

In Cassandra, we have retrieved all the data using region number and sequence id from the human_genome table. We have to use ALLOW FILTERING syntax to permit performing query by a particular field. Listing 7 represents this Cassandra query.

```
SELECT * FROM human_genome WHERE sequence CONTAINS
{'id':'3'} AND region_number= 4 ALLOW FILTERING;
```

Listing 7. Query performed in Cassandra

The query syntax of MongoDB is simpler between this three DBMS. MySQL query syntax is more complicated than the other two. Moreover, for Cassandra, we have to use ALLOW FILTERING for retrieving data. We have compared data query time for MySQL, MongoDB, and Cassandra in milliseconds. We have conducted four different queries and finally calculated the average for each. We perform four queries similar to the above listings. Fig. 4 shows the query response time. Cassandra shows the best performance in the retrieval operations. The MySQL shows worse results, and MongoDB also shows similar performance that of Cassandra. Therefore, Cassandra and MongoDB are preferable for large scale unstructured data, even if the data is made structured. Table I shows the improvement comparison of the result for data load time and query response time. NoSQL databases perform faster than MySQL on a single data entity in the case of a read or write operation. Because of the dynamic schema, the NoSQL database provides better scalability and flexibility for extensive unstructured data.

TABLE I
IMPROVEMENT COMPARISON BETWEEN THE DBMS

Comparison	Better	Data Load Time	Query Response Time
MongoDB vs Cassandra	Cassandra	77.63%	11.42%
MySQL vs Cassandra	Cassandra	90.48%	43.62%
MySQL vs MongoDB	MongoDB	57.45%	36.35%

C. Query for Sequence Matching

Matching the genome sequence is an important and practical operation for genomic data. In this section, we show the performance for genome sequence data matching using a random sequence. For example, we find all occurrences of a particular sequence (say *'AGGAGGAATGCTACAA...'*) in the dataset. We perform 4 similar sequence matching queries as in Listing 5 - Listing 7. Fig. 5 shows the query response time. Once again, Cassandra shows the best performance among the DBMS systems.

VI. CONCLUSION

In this paper, we analyzed the query performance of tremendous amounts of datasets of the human genome data from the COVID-19 dataset in SQL and NoSQL DBMS. We used MySQL for SQL DBMS and Cassandra and MongoDB for NoSQL DBMS. We describe the genomic domain along with its unstructured nature of data. We show a comparison in the case of data load time where Cassandra performs well. Moreover, we represent the different query syntax, and in the case of

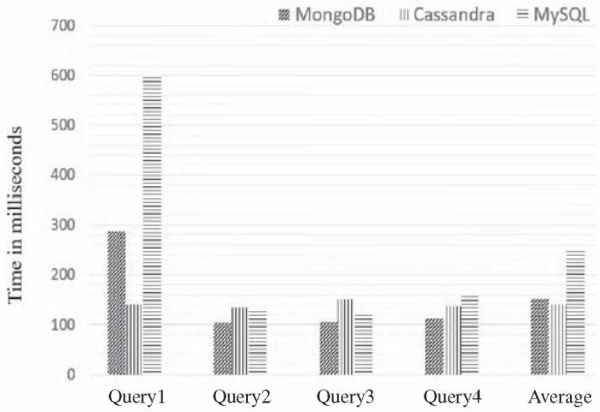

Fig. 5. Query response time for sequence matching (in milliseconds)

MySQL, we have to join two tables, and its operations become extensive. On the other hand, MongoDB's query seemed pretty straightforward, small, and easy to operate. Besides, its performance was also competitive. Cassandra's performance shows the best among the three query performance, and its query syntax is also simple. Finally, we analyzed four different queries on each of this DBMS, and it is cleared that Cassandra and MongoDB performed well comparing to MySQL in the case of unstructured data. In the future, this work can be extended to compare aggregate function in the case of a read, write, and delete as these operations are fundamental to any DBMS. Moreover, it is also essential to compare database operations for the application design stage and at regular intervals to enable switching to the most suitable database implementation for unstructured data. Therefore, for future work, we will analyze several unstructured data sets for its scalability and consistency issues using NoSQL models such as MongoDB, Cassandra, and also for other types of NoSQL DBMS like Hbase, HIVE, CouchDB to select a NoSQL DBMS for specific industrial applications.

REFERENCES

[1] A. Lakshman and P. Malik, "Cassandra: a decentralized structured storage system," *ACM SIGOPS Operating Systems Review*, vol. 44, no. 2, pp. 35–40, 2010.

[2] J. Dean and S. Ghemawat, "Mapreduce: simplified data processing on large clusters," *Communications of the ACM*, vol. 51, no. 1, pp. 107–113, 2008.

[3] P. Zikopoulos, C. Eaton *et al.*, *Understanding big data: Analytics for enterprise class hadoop and streaming data*. McGraw-Hill Osborne Media, 2011.

[4] G. DeCandia, D. Hastorun, M. Jampani, G. Kakulapati, A. Lakshman, A. Pilchin, S. Sivasubramanian, P. Vosshall, and W. Vogels, "Dynamo: amazon's highly available key-value store," *ACM SIGOPS operating systems review*, vol. 41, no. 6, pp. 205–220, 2007.

[5] F. Chang, J. Dean, S. Ghemawat, W. C. Hsieh, D. A. Wallach, M. Burrows, T. Chandra, A. Fikes, and R. E. Gruber, "Bigtable: A distributed storage system for structured data," *ACM Transactions on Computer Systems (TOCS)*, vol. 26, no. 2, pp. 1–26, 2008.

[6] K. Chodorow, *MongoDB: the definitive guide: powerful and scalable data storage*. " O'Reilly Media, Inc.", 2013.

[7] J. Wang, W. Wang, R. Li, Y. Li, G. Tian, L. Goodman, W. Fan, J. Zhang, J. Li, J. Zhang *et al.*, "The diploid genome sequence of an asian individual," *Nature*, vol. 456, no. 7218, pp. 60–65, 2008.

[8] D. Pushkarev, N. F. Neff, and S. R. Quake, "Single-molecule sequencing of an individual human genome," *Nature biotechnology*, vol. 27, no. 9, pp. 847–850, 2009.

[9] D. A. Wheeler, M. Srinivasan, M. Egholm, Y. Shen, L. Chen, A. McGuire, W. He, Y.-J. Chen, V. Makhijani, G. T. Roth *et al.*, "The complete genome of an individual by massively parallel dna sequencing," *nature*, vol. 452, no. 7189, pp. 872–876, 2008.

[10] A. Friends, "Xampp apache+ mariadb+ php+ perl," *Apache Friends*, 2017.

[11] R. Karimi, L. Bellatreche, P. Girard, A. Boukorca, and A. Hajdu, "Binos4dna: Bitmap indexes and nosql for identifying species with dna signatures through metagenomics samples," in *International Conference on Information Technology in Bio-and Medical Informatics*. Springer, 2014, pp. 1–14.

[12] B. Vela, J. M. Cavero, P. Cáceres, A. Sierra-Alonso, and C. E. Cuesta, "Using a nosql graph oriented database to store accessible transport routes." in *EDBT/ICDT Workshops*, 2018, pp. 62–66.

[13] E. Estellés-Arolas and F. González-Ladrón-De-Guevara, "Towards an integrated crowdsourcing definition," *Journal of Information science*, vol. 38, no. 2, pp. 189–200, 2012.

[14] B. D O'Connor, B. Merriman, and S. F. Nelson, "Seqware query engine: storing and searching sequence data in the cloud," in *BMC bioinformatics*, vol. 11, no. S12. Springer, 2010, p. S2.

[15] M. Banane, A. Belangour, and L. El Houssine, "Storing rdf data into big data nosql databases," in *First International Conference on Real Time Intelligent Systems*. Springer, 2017, pp. 69–78.

[16] I. Konstantinou, E. Angelou, C. Boumpouka, D. Tsoumakos, and N. Koziris, "On the elasticity of nosql databases over cloud management platforms," in *Proceedings of the 20th ACM international conference on Information and knowledge management*, 2011, pp. 2385–2388.

[17] Z. Parker, S. Poe, and S. V. Vrbsky, "Comparing nosql mongodb to an sql db," in *Proceedings of the 51st ACM Southeast Conference*, 2013, pp. 1–6.

[18] Y. Li and S. Manoharan, "A performance comparison of sql and nosql databases," in *2013 IEEE Pacific Rim Conference on Communications, Computers and Signal Processing (PACRIM)*. IEEE, 2013, pp. 15–19.

[19] E. Tang and Y. Fan, "Performance comparison between five nosql databases," in *2016 7th International Conference on Cloud Computing and Big Data (CCBD)*. IEEE, 2016, pp. 105–109.

[20] J. C. Wooley, A. Godzik, and I. Friedberg, "A primer on metagenomics," *PLoS Comput Biol*, vol. 6, no. 2, p. e1000667, 2010.