Efficient data structures for high-volume time-series bridge sensor data

Andrew Holmberg, Marcello Balduccini

Department of Decision and System Sciences, Saint Joseph's University, Philadelphia, United States

{andrew.holmberg, mbalducc}@sju.edu

Abstract. This paper evaluates several data models for high-volume time-series bridge sensor data. Remote sensor technologies are becoming increasingly more reliable and affordable for ensuring safety and guaranteeing appropriate timing for bridge maintenance. Data returned from these sensors tends to be of a very high volume, with modern sensors returning hundreds of readings per second. When working with such a large volume of data, concerns arise with how efficiently the data can be ingested and retrieved. Inefficient data ingestion can cause the data transportation system to be overwhelmed and incorrect or incomplete data to be pushed to the database. Inefficient retrieval will limit the ability of bridge stakeholders to make real-time data-driven decisions about the safety of bridges. Data structure becomes a critical part of ensuring that these two operations can be performed as fast as possible with the technology available. This paper reviews the top current technologies for managing both structured and semi-structured data through the lens of high-volume time-series data from bridge sensors. We propose several potential structured and semi-structured data models for bridge sensor data and implement them with the appropriate technologies for each. The proposed data models are tested using data collected from actual bridges. We conclude by recommending a semi-structured data model that allows for data to be both more easily collected and ingested, without making any concessions in terms of the speed at which the data can be retrieved for data visualizations and analysis.

1. Introduction

In the environment of the Internet of Things, bridges are increasingly being monitored with remote sensors to ensure safety and guarantee appropriate timing for bridge maintenance. With this, effective sensor data ingestion and retrieval has become one of the main challenges in successful monitoring of bridges. Bridge monitoring using modern sensors will return hundreds or thousands of readings per second from each individual sensor [1]. This brings challenges in both the storage of such a large quantity of data and the retrieval of the data to make informed decisions about bridge safety. Conclusions cannot be easily drawn from such a substantial amount of sensor data, meaning that any analysis requires effective data analysis procedures [2]. As a result, resources such as data visualization tools and machine learning algorithms are needed to help gather meaningful information from the data returned [3] [4]. However, efficient retrieval of sensor data for these processes becomes increasingly important as the volume of data becomes larger [5]. The volume of data being processed will further intensify if there is a need to also store video files to accompany sensor data in the data visualizations.

There are several data models that high-volume time-series sensor data can fit into. A data model with a tabular structure can be chosen to make use of all the technology associated with relational databases. Since structured data predates semi-structured data, the traditional ways of storing and analyzing data are built to support structured data [6]. Other times, a semi-structured data model for sensor data will be deemed appropriate. Semi-structured data does not need to confine itself to the tabular structure of relational databases, but instead can use certain markers to separate elements and enforce hierarchies [7]. JSON objects are a viable semi-structured data model for sensor data and will be the semi-structured data model that is the focus of this paper [8]. The popularity of JSON objects as a data model and the increasing need for managing large volumes of data has led to the creation of database technologies that have the native ability to work with data in JSON format [9]. The large volume that sensor data is collected benefits from semi-structured data models not obligating that data be strictly typed or predefined [7]. Sensor data does not inherently lend itself to a semi-structured or structured data model, and the choice is made on a case-by-case basis for the particular needs of the data manager.

When working with bridge sensor data there are two needs that are immediately present. The sensor data needs to be ingested efficiently to prevent the data transportation system from being overwhelmed and causing inaccurate or incomplete data to be pushed to the database. With data concerning the safety of bridges, the accuracy of data is paramount to ensure correct analyses and visualizations. The second need is to be able to efficiently query the store of data to quickly assemble visualizations and perform machine learning algorithms. Making worthwhile conclusions about large amounts of bridge sensor data is not a trivial task. As such, these methodologies will need to be performed quickly and often to make meaningful bridge management decisions from the sensor data.

In this paper, we assess various data models suitable for handling high-volume time-series data from bridge sensors. Our evaluation focuses on current technologies that can manage both structured and semi-structured data. We propose and implement structured and semi-structured data models, using the appropriate technologies for each. Our proposed data models are tested using actual bridge sensor data, and we conclude by recommending a semi-structured data model that facilitates easier data collection, ingestion, and retrieval.

The paper is organized as follows. In the next section, we discuss the criteria used for selecting database technologies chosen to test our data models. In Section 3, we propose both structured and semi-structured data models that could be used for high-volume time-series sensor network data collection. Section 4 covers ingestion efficiency and Section 5 covers retrieval efficiency of the proposed data models and database technologies with real sensor measurements and image data from bridges. The final section draws conclusions and outlines future work.

2. Database Technologies Considered

To benchmark the performance of our proposed data models, database technologies needed to be chosen for both structured and semi-structured data models. When selecting database technology for high-volume time-series sensor data several considerations must be taken into account. The database technology should be able to provide a guarantee of data integrity and consistency, particularly in the face of high write loads. In addition, performance is also a critical factor. The database technology needs to be able to handle high write and read throughput, with read throughput being particularly essential for real-time monitoring and analysis of sensor data. Lastly, the database technology must be optimized for time-series data and support time-based querying. This includes fetching all data for a given time range, as well as other time-based queries that may be necessary. Within these criteria, only open-source database technologies were considered as their cost, transparency, security, and flexibility make them a suitable option for the widest range of use cases.

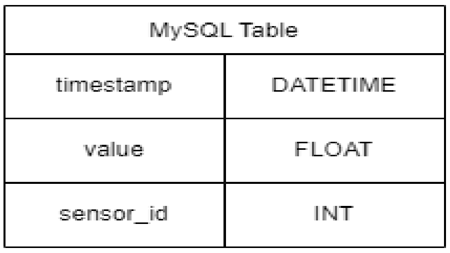
PostgreSQL and MySQL are the two database technologies chosen for benchmarking the structured data models. Both PostgreSQL and MySQL are open-source and offer support for data integrity through constraints [10] [11] and locking [12] [13]. In addition, both PostgreSQL and MySQL provide the ability to query on time intervals [14] [15]. In comparative benchmarking of relational databases, PostgreSQL and MySQL also both outperform other commercial database technologies, particularly with large data objects [16].

For benchmarking the semi-structured data models, MongoDB and Elasticsearch are the two chosen database technologies. MongoDB and Elasticsearch are also both open-source technologies and ensure data integrity through locking [17] [18]. Unlike traditional structured databases, which require strict adherence to a predefined schema, semi-structured databases allow for more flexibility and dynamic data modeling [9]. Constraints play less of a role in semi-structured databases because there is no predefined schema to constrain the data to. If it would benefit a data manager to have a predefined schema, Elasticsearch does offer the ability to define a document's fields ahead of time through mapping [19]. In comparative benchmarking, MongoDB is noted as having superior execution time compared to the top two other semi-structured database technologies, CouchDB and Couchbase [9]. The key feature in terms of performance for Elasticsearch is its searching, which can be done on a variety of constraints and at near real-time [20]. The ability to query on time intervals is present in both MongoDB and Elasticsearch.

3. Data Models

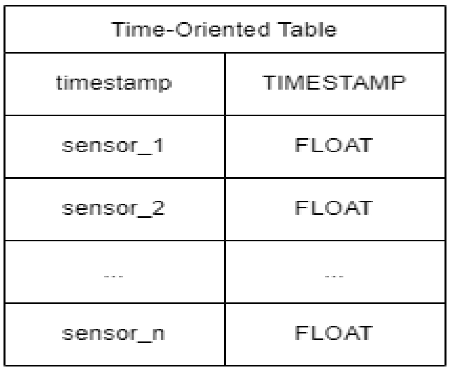
For this paper, each record of data collected from a sensor network carries three pieces of information. These are timestamps that indicate when the record was observed, the values measured by the sensor at that time, and some value indicating the specific sensor from the network that recorded the measurement. To ingest this data in a column-oriented database each of these fields takes on one column, with the timestamp column holding TIMESTAMP values in PostgreSQL and DATETIME values in MySQL, the value column holding FLOAT values in both PostgreSQL and MySQL, and the sensor_id column holding INT values in both PostgreSQL and MySQL.

The largest piece of data contained in these two data models is the timestamp value. This means that the timestamp will also be the costliest in terms of retrieval and ingestion. Therefore, an effort can be made to improve the efficiency of the data model by minimizing the number of times that a given timestamp must be ingested and retrieved. Modern sensor networks offer the ability to perform synchronized network sampling [1]. In synchronized network sampling, all the sensors in the network record measurements at precisely the same timestamp value. A possible alternate data structure for synchronized sampling networks is shown in figure 2.



MySQL Table	
timestamp	DATETIME
value	FLOAT
sensor_id	INT

Figure 1. Proposed structured data model to be used with MySQL

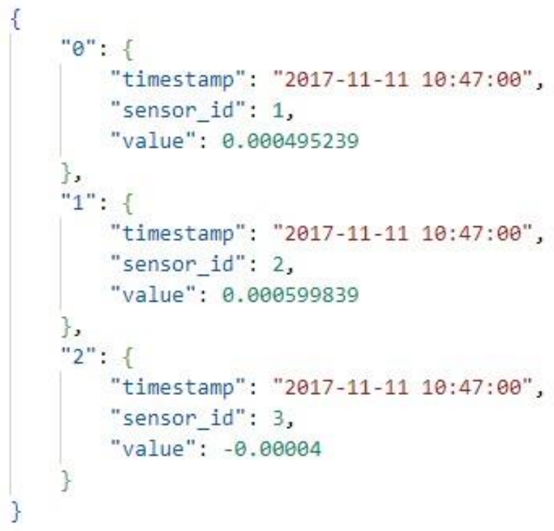


Time-Oriented Table	
timestamp	TIMESTAMP
sensor_1	FLOAT
sensor_2	FLOAT
...	...
sensor_n	FLOAT

Figure 2. Possible time-oriented structured data model for a synchronized sampling network

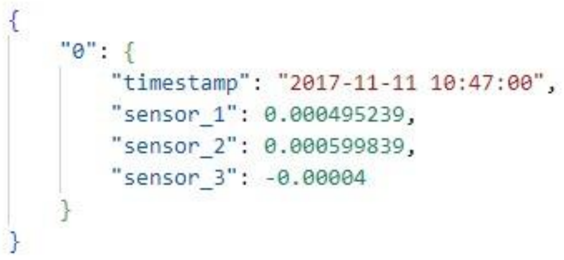
In this alternate data model, only unique timestamp values are stored, and each sensor on the network has its own column designated for it. While this data model would decrease the overall quantity of data that needs to be stored, it has limited feasibility for a practical sensor network. Structured data models require a data manager to predefine the schema for the data model ahead of time. One can expect a real high-volume wireless sensor network to dynamically add or drop sensors from the network as time goes on. This would make the data model illustrated in figure 2 infeasible because the schema cannot dynamically change to reflect sensors being added or dropped from the network.

Semi-structured data models benefit in that they do not need to predefine a schema ahead of time. The JSON objects could be designed in a similar style to figure 1, with each sensor reading being associated with its own timestamp, measured value, and sensor identifier as shown in figure 3. This will be referred to as sensor orientation. In addition, the structured data model depicted in figure 2 also becomes feasible in a semi-structured approach. Each JSON object could hold a single timestamp but hold every sensor reading from a synchronized sampling network from that moment. If sensors are dynamically added or dropped from the network, this will not be an issue because there is no predefined schema dictating what fields must exist in the JSON object. This will be referred to as time orientation.



```
{
    "0": {
        "timestamp": "2017-11-11 10:47:00",
        "sensor_id": 1,
        "value": 0.000495239
    },
    "1": {
        "timestamp": "2017-11-11 10:47:00",
        "sensor_id": 2,
        "value": 0.000599839
    },
    "2": {
        "timestamp": "2017-11-11 10:47:00",
        "sensor_id": 3,
        "value": -0.00004
    }
}
```

Figure 3. Semi-structured implementation of a sensor-oriented data model



```
{
    "0": {
        "timestamp": "2017-11-11 10:47:00",
        "sensor_1": 0.000495239,
        "sensor_2": 0.000599839,
        "sensor_3": -0.00004
    }
}
```

Figure 4. Semi-structured implementation of a time-oriented data model

In considering the image data that may be necessary for visualizations or analysis, this paper will also test the ability of databases to ingest and retrieve large binary objects. Both PostgreSQL and MySQL can natively store this sort of data directly in the database as type BYTEA and BLOB respectively. Large binary objects can also be mapped in an Elasticsearch index as type BINARY. To test ingestion and retrieval, the image data is stored alongside the corresponding timestamp.

4. Testing Ingestion Efficiency

This section of the paper discusses the testing methodology and results for measuring the efficiency of data ingestion of our chosen database technologies and proposed data models. All experimental data and docker files are available at [21]. All testing for ingestion and retrieval efficiency was done on a 2.3 GHz server with 24GB main memory. The operating system used was a CentOS Linux 7 (Core).

In an implementation of a sensor network, additional strain will be put on the data infrastructure for each additional sensor added to the network. This makes it important to test not only how efficiently the data pipeline can ingest data from a single sensor, but also how efficiently data can be ingested from several sensors from the same time frame.

To accomplish this, the testing was done with real sensor measurements of displacement that were collected from an actual bridge by six different sensors at the same time. The tests measured the average amount of time it took to ingest one second of sensor data, and how that amount of time increased as the number of sensors on the network increased. The sensors used measured displacement at a frequency of 128 measurements per second. For each data structure, the average time for ingestion over 100 trials was considered. Tests were done for the structured data models using both PostgreSQL and MySQL, and tests were done for the semi-structured data models using both MongoDB and Elasticsearch.

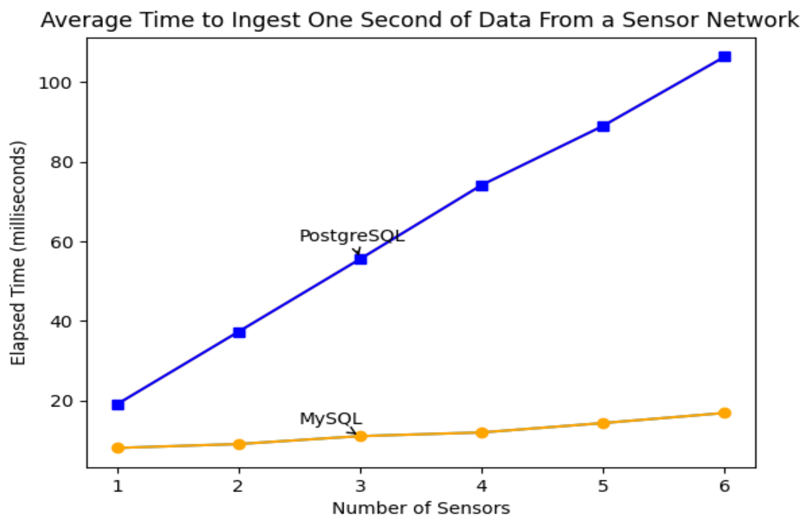


Figure 5. Milliseconds to ingest one second of data as a function of the number of sensors on the network for the structured data model

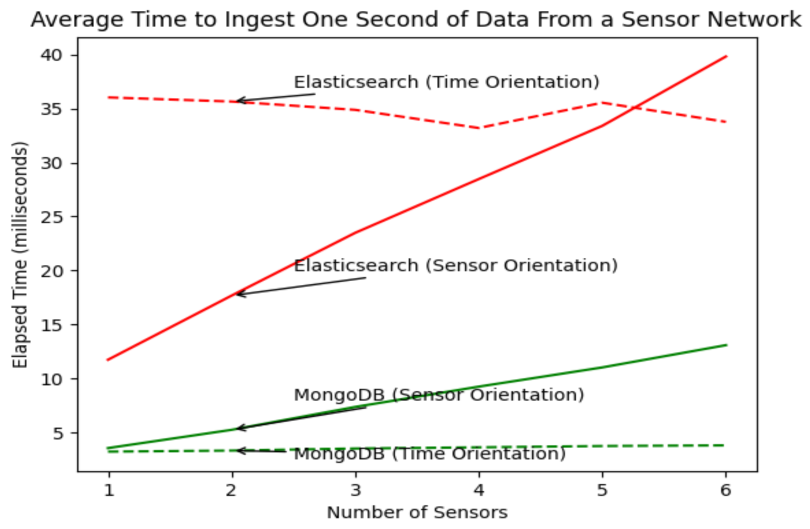


Figure 6. Milliseconds to ingest one second of data as a function of the number of sensors on the network for the semi-structured data models

For the proposed structured data model, MySQL outperforms PostgreSQL in terms of data ingestion. For both proposed semi-structured data models, MongoDB outperforms Elasticsearch in terms of data ingestion. The significant difference between the performance of the time orientation compared to the sensor orientation for MongoDB can be explained by the fact that less timestamps, the costliest part of the sensor data, are being ingested. The time orientation has only one timestamp ingested per every six sensor readings, while as the sensor orientation ingests a timestamp value for every sensor reading. Elasticsearch does not see a similar boost in performance from using the time orientation because it prohibits Elasticsearch from using its mapping feature. Explicitly defining the schema for the data can help Elasticsearch optimize the ingestion process. For example, defining a field as a timestamp beforehand can allow Elasticsearch to optimize using the date histogram facet [22]. Like MongoDB, Elasticsearch has very little ingestion cost associated with adding an additional sensor to an already existing synchronized sampling network.

The ingestion of image data was tested by inserting one second of video footage of a bridge into each of the four database technologies. The video was recorded at thirty frames per second, so this entailed ingesting thirty large binary objects alongside their corresponding timestamp. This test was performed 100 times and the average of those times are shown in table 1.

Table 1. The average amount of time in milliseconds to ingest one second of a video of a bridge recorded at 30 frames per second

Database Technology	Elapsed Time (milliseconds)
MySQL	136.28
PostgreSQL	177.83
MongoDB	79.03
Elasticsearch	476.24

5. Testing Retrieval Efficiency

The retrieval tests were done on the database technologies with one hour of data from a six-sensor synchronized sampling network already ingested. The data used for retrieval is an extension of the same dataset that was used to test ingestion. Python code was used to generate 100 random fifteen second intervals within the one hour of available data. For each interval, the sensor readings from that interval were retrieved, sorted in chronological order, and stored in a Python array. This process was used to mimic how the data might be retrieved from a database to create a visualization to analyze and interpret the data.

The average amount of time for these operations was noted across the one hundred intervals. Similarly to the methodology used in the ingestion testing, testing was also done to see how the amount of time required to retrieve the data changed as the number of sensors that the visualization required increased. The amount of data points that needed to be retrieved for each trial was equal to 1920, the number of readings every fifteen seconds for each sensor, multiplied by the number of sensors that data was being retrieved for.

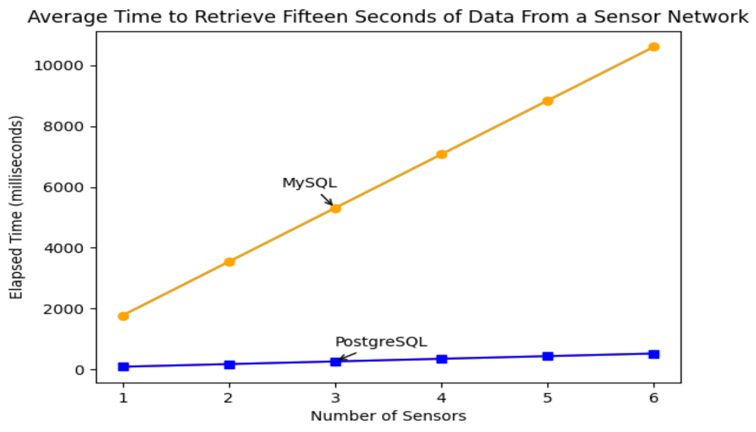


Figure 5. Time to retrieve as a function of the number of sensors on the visualization for structured data models

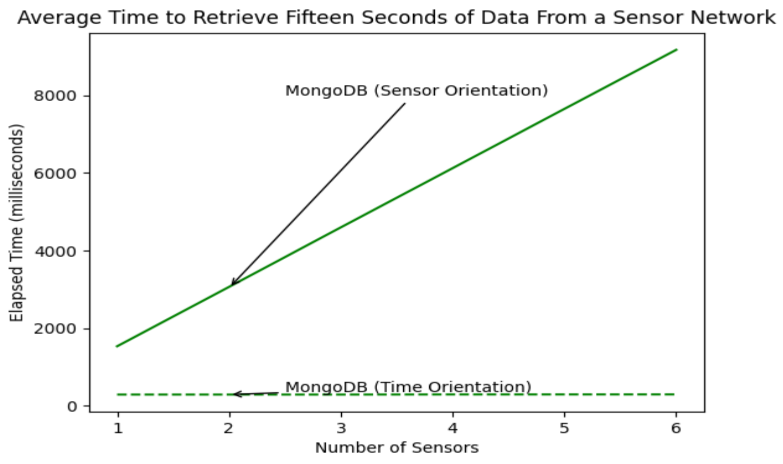


Figure 5. Time to retrieve as a function of the number of sensors on the visualization for semi-structured data models on MongoDB

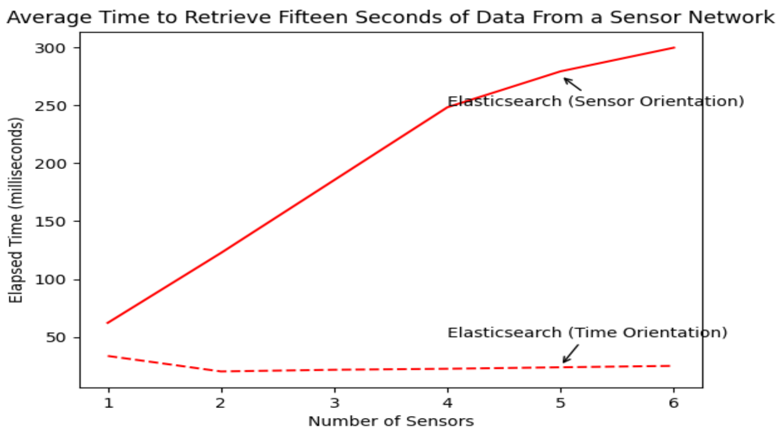


Figure 5. Time to retrieve as a function of the number of sensors on the visualization for semi-structured data models on Elasticsearch

For the proposed structured data model, PostgreSQL outperformed MySQL in terms of data retrieval. The time-oriented semi-structured data model outperformed the sensor-oriented semi-structured data model across both MongoDB and Elasticsearch. Adding an additional sensor to the visualization produced approximately linear growth in the amount of time it took to retrieve the data for the visualization in both the structured data model and the sensor-oriented semi-structured data model. In both cases, additional records needed to be retrieved to add additional sensors. In the case of the time-oriented semi-structured data model, only one record needed to be achieved for each timestamp value, no matter how many sensors were being used for the visualization. Also with time orientation, there is the benefit of there being fewer total records in the database that need to be searched. With the optimal time-oriented semi-structured data model, Elasticsearch outperformed MongoDB in terms of data retrieval.

The retrieval of image data was tested by retrieving fifteen seconds of video footage of a bridge from each of the four database technologies. The intervals used were the same fifteen second intervals that were generated to test the sensor retrieval. The average time to retrieve the images constituting fifteen seconds of video across the one hundred intervals is shown in table 2.

Table 2. The average amount of time in milliseconds to retrieve fifteen seconds of video of a bridge recorded at 30 frames per second from a database containing one hour of the footage

Database Technology	Elapsed Time (milliseconds)
MySQL	324.04
PostgreSQL	407.11
MongoDB	286.51
Elasticsearch	1358.53

6. Conclusion

Effective bridge monitoring using remote sensors requires efficient sensor data ingestion and retrieval to ensure safety and timely maintenance. In this paper, we evaluated a number of structured and semi-structured data models and of database technologies in order to identify the most suitable combination for this safety-critical domain. High-volume time-series sensor data can fit into either a structured or semi-structured data model, with JSON objects being a viable option for the latter. The open-source database technologies PostgreSQL, MySQL, MongoDB, and Elasticsearch were chosen for benchmarking the performance of a structured data model, a sensor-oriented semi-structured data model, and a time-oriented semi-structured data model. Our testing found the time-oriented semi-structured data model to be superior in both ingestion and retrieval. In addition, we evaluated the ability of the four database technologies to hold video footage of bridges in the form of large binary objects. MongoDB performed the best out of the four database technologies tested in both ingestion and retrieval. Future research may include the efficiency of data retrieval for these data models and database technologies for data analysis by machine learning algorithms.

Acknowledgements
Portions of this publication and research effort are made possible through the support of Federal Highway Administration via cooperative agreement 693JJ32150007.

References

[1] Furkan M O, Mao Q, Livadiotis S, Mazzotti M, Aktan A E, Sumitro S P and Bartoli I 2020 Towards rapid and robust measurements of highway structures deformation using a wireless sensing system derived from wired sensors *Journal of Civil Structural Health Monitoring* **10** 297-311

[2] Rajawat A S, Bedi P, Goyal S B, Alharbi A R, Aljaedi A, Jamal S S and Shukla P K 2021 Fog big data analysis for IoT sensor application using fusion deep learning *Mathematical Problems in Engineering*

[3] Plötz T 2021 Applying machine learning for sensor data analysis in interactive systems *ACM Computing Surveys* **54**

[4] Balakrishna S, Thirumaran M and Solanki V 2018 A framework for IOT sensor data acquisition and analysis *EAI Endorsed Transactions on Internet of Things* **4**

[5] Dou J, Chu L, Cao J, Qiu Y and Zhao B L 2020 Efficient optimized strategy of big data retrieval *Proc. of the 2020 6th Int. Conf. on Computing and Artificial Intelligence* 109-116

[6] Elgendy N and Elragal A 2014 Big data analytics: a literature review paper *Industrial Conf. on Data Mining* 214-227

[7] Dickson M and Asagba P O 2020 The semi-structured data model and implementation issues for semi-structured data *International Journal of Innovation and Sustainability* **3** 47-51

[8] Antolín D, Medrano N, Calvo B and Pérez F 2017 A wearable wireless sensor network for indoor smart environment monitoring in safety applications *Sensors* **17**

[9] Carvalho I, Sá F and Bernardino J 2023 Performance evaluation of NoSQL document databases: Couchbase, CouchDB, and MongoDB *Algorithms* **16**

[10] https://dev.mysql.com/doc/refman/8.0/en/constraints.html

[11] https://www.postgresql.org/docs/current/ddl-constraints.html

[12] https://www.postgresql.org/docs/current/explicit-locking.html

[13] https://dev.mysql.com/doc/refman/8.0/en/innodb-locking.html

[14] https://dev.mysql.com/doc/mysql-tutorial-excerpt/8.0/en/selecting-rows.html

[15] https://www.postgresql.org/docs/current/rangetypes.html

[16] Stancu-Mara S and Baumann P 2008 A comparative benchmark of large objects in relational databases *12th Int. Database Engineering and Applications Symp.*

[17] https://www.mongodb.com/docs/manual/faq/concurrency/

[18] https://www.elastic.co/guide/en/elasticsearch/reference/current/optimistic-concurrency-control.html

[19] https://www.elastic.co/guide/en/elasticsearch/reference/current/mapping.html

[20] Kathare N, Reddy O V and Prabhu V 2021 A comprehensive study of Elasticsearch *International Journal of Science and Research* **10**

[21] https://github.com/Dr-B-AskLab/testing-bridge-sensor-data-models

[22] https://www.elastic.co/guide/en/elasticsearch/reference/2.3/search-aggregations-bucket-datehistogram-aggregation.html.