Sandra WŁOSTOWSKA[1], Julia SZABELA[2], Adrian CHOJECKI[1], Piotr BORKOWSKI[1]

Department of Electrical Apparatus, Faculty of Electrical, Electronics, Computer and Control Engineering, Lodz University of Technology (1)
Faculty of Organization and Management, Lodz University of Technology (2)

doi:10.15199/48.2023.11.02

Comparison of SQL, NoSQL and TSDB database systems for smart buildings and smart metering applications

Abstract. *The article compares different types of databases for storing data sourced from smart building and smart metering systems. The differences between non-relational (NoSQL – NoRDBS), time-series (TSDB) and the most popular relational database (SQL – RDBS) were highlighted. Then, the possibilities of practical usage of them for gathering data from typical sensors used in buildings and the integration with the most popular installation standards (KNX, LCN) and communication protocols (Modbus, MQTT, MBus) were examined. For examination, the most popular software under open-source licenses were selected: PostgreSQL as a relational database; MongoDB as a non-relational document database; and InfluxDB as a TSDB database. Installation and configuration methods, the possibility of integration with other systems and interfaces to popular programming languages (C#, Python, Java, C++) were compared. In the empirical part, the following were studied: access times and increase in used disk space in cooperation with a real sensor to reach 1 million records in each of the databases. In the conclusions, the results are presented and the authors' suggestions for potential applications for the databases in the fields mentioned in the title are provided.*

Streszczenie. *W artykule porównano różne typy baz danych na potrzeby przechowywania danych pochodzących z systemów inteligentnego budynku i smart-meteringu. Zwrócono uwagę na różnice pomiędzy bazami nierelacyjną (NoSQL – NoRDBS), opartą o szeregi czasowe (TSDB) i najpopularniejszą bazą relacyjną (SQL – RDBS). Następnie przeanalizowano możliwości ich praktycznego wykorzystania na potrzeby gromadzenia danych z typowych czujników stosowanych w inteligentnym budownictwie oraz integracji z najpopularniejszymi standardami instalacji (KNX, LCN) i protokołów komunikacyjnych (Modbus, MQTT, MBus). Do przebadania wybrano najpopularniejsze systemy na licencjach open-source: PostgreSQL jako baza relacyjna; MongoDB jako baza nierelacyjna dokumentowa oraz InfluxDB jako baza typu TSDB. Porównano sposoby instalacji i konfiguracji, możliwość integracji z innymi systemami oaz interfejsy dostępowe dla popularnych języków programowania (C#, Python, Java, C++), a w części empirycznej czasy dostępu i przyrost zajętej przestrzeni dyskowej we współpracy z rzeczywistym czujnikiem do osiągnięcia 1 miliona rekordów w każdej z baz. We wnioskach przedstawiono wyniki oraz zebrano sugestie autorów dotyczące potencjalnych zastosowań dla przetestowanych systemów baz danych w dziedzinach zawartych w tytule.* ***(Porównanie systemów baz danych SQL, NoSQL i TSDB dla zastosowań w inteligentnych budynkach i smart metering)***

Keywords: non-relational database, time series database, database systems, smart metering
Słowa kluczowe: nierelacyjna baza danych, baza danych szeregów czasowych, systemy bazodanowe, smart-metering

Introduction

Development of technology, falling prices of advanced electronic systems used in many fields and the growing number of sensors, as well as the development of machine learning algorithms, are enabling the benefits offered by the data.

A database is defined as any method of data storage, in a organised form, allowing data to be read, filtered, searched, etc. From the point of view of computer science and most fields of technology where they are used, it should be defined as a collection of information that is stored electronically.

Nowadays, there are many types of database systems available on the market, which differ in data structure and storage methods [1, 2]. A computer system that handles the database (Database Management System – DBMS) should support all operations related to the stored data, according to the C.R.U.D. acronym: C – Create, R – Read, U – Update, D – Delete [2]. This ensures transparency of the system, ease and comfort in use and improvement of the database creation process [1]. The most widespread systems up to now were Relational Database Systems (RDBS), using the SQL (Structured Query Language) and A.C.I.D. rule [3], ensuring data integrity during the transaction. According to this rule, operations performed on the database will be performed completely or canceled. The transaction is persistent and performed independently of others, and the system remains consistent after its execution.

However, the explosion in the number of data sources, as well as the emergence of big data conventions and the occurring changes in the structure and type of data stored during the system's operation have led to an interest in other types, than the most widespread RDBS, which will nevertheless remain in practical use for a long time to come.

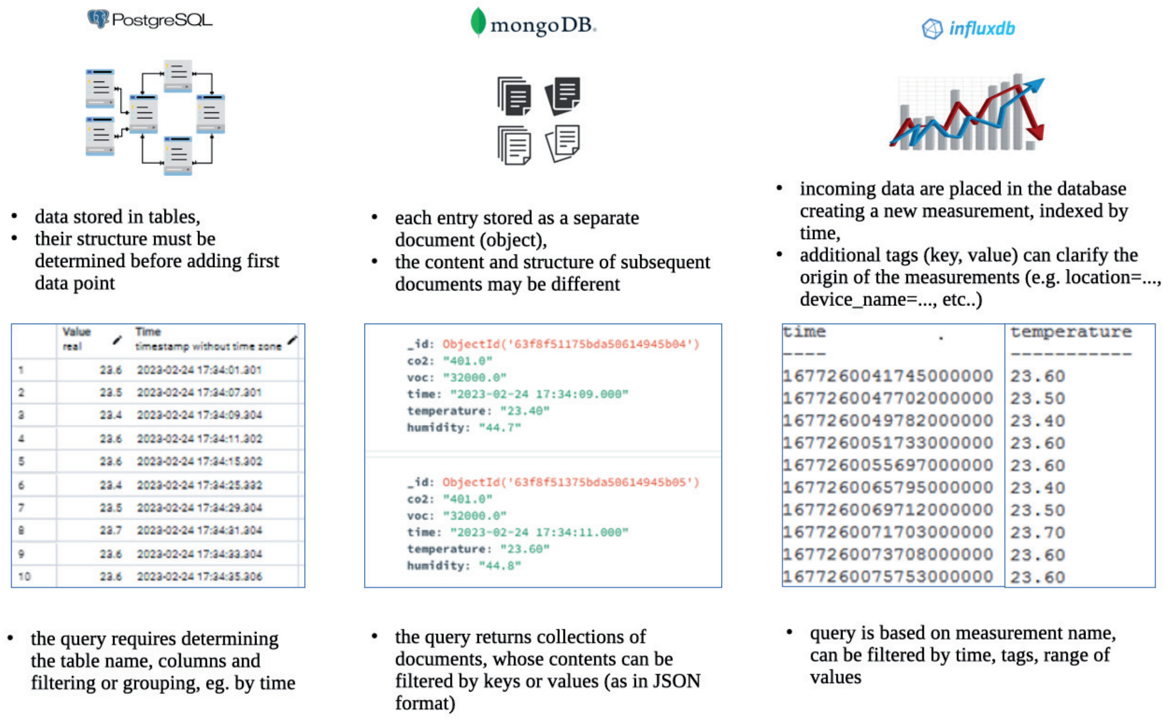
The paper compares different types of databases for storing data sourced from smart building and smart metering systems. Its important aspect was addressing the problem of database implementation in the fields of metrology, allowing for the creation of intelligent measurement systems, which were referred to as smart metering. For examination, popular open-source software was selected: PostgreSQL [4] as a relational database with SQL, MongoDB [5] as a non-relational document database, and InfluxDB [6] as a TSDB (Time Series Database). Installation, configuration methods, integration possibilities with other systems, and interfaces to popular programming languages (C#, Python, Java, C++) were compared. In the main part of the study, a sample device with DHT21 [7] and SGP30 [8] air-quality sensors connected to the OrangePI Zero® [9] Single Board Computer (SBC) sent data to the examined DBMS to compare response times and disk space utilization for up to 1 million records. Finally, the practical applications of the investigated DBMS were discussed, and conclusions were presented.

Smart buildings and smart metering

The term smart metering is used to describe an intelligent metering system which, in addition to measuring energy consumption, also allows you to obtain even more valuable information than is the case with a conventional meter [10]. This concept mainly refers to Advanced Metering Infrastructure (AMI), an element of smart-grid and will be provided on the grid operator's side, databases are a key component of such systems, however, this applies on a slightly different level [11]. A comparative assessment of DBMS for smart metering purposes on the grid side, including also the databases explored in this paper, was presented by Olivares-Rojas et al. in [12].

On the other hand, all systems installed individually by the end-consumers and those being a component of Building Automation and Controls Systems (BACS) and integrated Building Management Systems (BMS), providing the ability to acquire measurement data, analyze it and provide insights also are classified as smart metering.

According with the new wording of *Directive EU on the energy performance of buildings* [13], measurement data collection is an important component of automation systems, al-



- data stored in tables,
- their structure must be determined before adding first data point

	Value real	Time timestamp without time zone
1	23.6	2023-02-24 17:34:01.301
2	23.5	2023-02-24 17:34:07.301
3	23.4	2023-02-24 17:34:09.304
4	23.6	2023-02-24 17:34:11.302
5	23.6	2023-02-24 17:34:15.302
6	23.4	2023-02-24 17:34:25.332
7	23.5	2023-02-24 17:34:29.304
8	23.7	2023-02-24 17:34:31.304
9	23.6	2023-02-24 17:34:33.304
10	23.6	2023-02-24 17:34:35.306

- the query requires determining the table name, columns and filtering or grouping, eg. by time

- each entry stored as a separate document (object),
- the content and structure of subsequent documents may be different

```
_id: ObjectId('63f8f51175bda50614945b04')
co2: "401.0"
voc: "32000.0"
time: "2023-02-24 17:34:09.000"
temperature: "23.40"
humidity: "44.7"

_id: ObjectId('63f8f51375bda50614945b05')
co2: "401.0"
voc: "32000.0"
time: "2023-02-24 17:34:11.000"
temperature: "23.60"
humidity: "44.8"
```

- the query returns collections of documents, whose contents can be filtered by keys or values (as in JSON format)

- incoming data are placed in the database creating a new measurement, indexed by time,
- additional tags (key, value) can clarify the origin of the measurements (e.g. location=..., device_name=..., etc..)

```
time                 temperature
----                 -----------
1677260041745000000  23.60
1677260047702000000  23.50
1677260049782000000  23.40
1677260051733000000  23.60
1677260055697000000  23.60
1677260065795000000  23.40
1677260069712000000  23.50
1677260071703000000  23.70
1677260073708000000  23.60
1677260075753000000  23.60
```

- query is based on measurement name, can be filtered by time, tags, range of values

Fig. 1: A short overview of the examined DBMS's features and example view of retrieved data.

lowing subsequent analysis, evaluation of energy efficiency and improvement of control algorithms and in the future this may even become a legislative obligation for European countries [14]. Collecting measurement data is also an important element for various IoT and air-quality control applications. The data provided by such data sources are obviously in the form of time series. A more general study, but mostly theoretical comparison of open source databases for time series data was presented by Bader et al. in [15]. Edge IoT applications of different DBMS for air quality measurement were presented by Grzesik et al. in [16].

Unfortunately, a lot of market available BACS and BMS solutions are proprietary systems and not always offering in their core version an easy interface for retrieving and analyzing the collected data. Also, often is still used the approach from classical industrial automation, with SCADA and OPC software. This is relevant and reliable for closed, proprietary production plants, but in the era of big-data and data analytic, for building automation may be a limitation or may require an additional layer of integration [17].

In all these cases, adopting an external database, may allow easier management of the acquired data, increase the reliability through isolation from the on-field layer and opens the doors to analyzing it using modern machine learning methods and user-friendly visualization tools like Grafana [18] and Kibana.

Examined databases

A short overview of the analyzed databases features and an example view of the collected data is shown in Fig. 1, while concise descriptions of each are given below.

PostgreSQL - Relational database

A relational database contains structured and interconnected information that exists in appropriate relationships. All records that will be stored will form a broader structure called a table and are linked between tables using foreign keys. One or more tables can exist in a single database [10].

Access to information is possible with a specific language for relational databases – SQL. The SQL language has become an industry standard and a similar syntax is also used in other types of DBMS [1, 6].

The PostgreSQL [4] was written in the C language and it is one of the few systems to offer an object-relational approach. It is available under an open source license and accessible to the majority of operating systems. The tool used to manage the PostgreSQL database is pgAdmin [19], one of the most popular tools which is running as web-application and open in browser. Users rate this tool the highest for code review and repository management, according to statistics, 75% of users will recommend PostgreSQL with PgAdmin [19, 20].

MongoDB - Non-relational, document database

The term NoSQL (Not only SQL) characterizes a certain group of modern database systems oriented toward use in distributed information systems. Such databases do not have a predefined structure, it is often created automatically when data is added. This is a useful feature when the data has a very diverse structure or when the structure is not immediately defined. In NoSQL there are no typical relationships, but there are some solutions that can simulate relationships known from RDMS. Queries can be created with scripting languages [5, 21].

One of the most popular [20] representatives of a non-relational database system is MongoDB [5], an open-source system developed in C++ language. Data is stored in the form of documents saved in JSON format, or BSON of the binary variety. Instead of tables, the database contains collections, performing similar function. Collections are characterized by the lack of a specific structure, they do not have standard columns, so the documents differ from each other. In a non-relational database, queries most often are created in JavaScript, but MongoDB offers drivers for many other languages. Due to the lack of relationships, collections do not

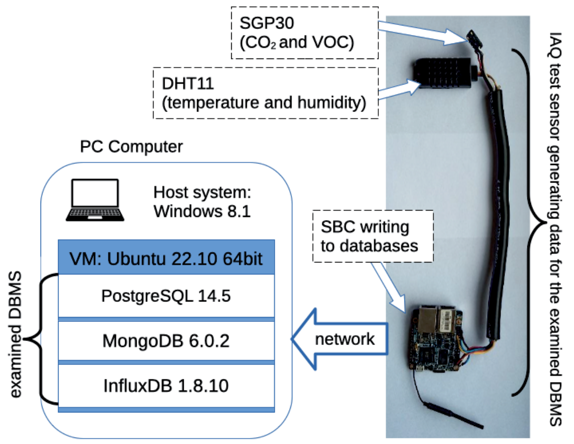


Fig. 2: Test setup for installation and real data collection.

link to each other, sometimes results in duplicate data. MongoDB has the advantage of speed in operation, which makes it possible to work with big-data [21].

InfluxDB - Time Series database

Time Series Database (TSDB) is defined as a software system adapted for storing data in form of time series, representing related pairs of timestamps and values. Time series data can be measurements or events aggregated over time, building a sequence of observations [6, 22]. Interest in TSDBs has grown significantly with the emergence of the IoT concept and the availability large amount of data from numerous sensors, fetched at certain time intervals.

InfluxDB [6] is an one of the most popular open-source software that supports a time series database. It was designed for tasks that generate large amounts of data over time, handles large workloads and queries well. The data query language is InfluxQL, with the structure resembles standard SQL. Data points consist of key-value pairs, the series of data points are then grouped together according to the string identifier called tags and form a measurement [23]. The newest releases introduced an additional, function-based query language flux, but InfluxQL is still used in many real-life applications [22].

Methodology

Once the DBMS software for the study was selected, based on rankings published in [20] and previous works such as [10, 11, 12, 15, 16], an analysis of functionality, installation methods, and suitability for the specified field was conducted. The chosen DBMSs, except for InfluxDB, ranked among the top 5 and InfluxDB was included due to its popularity in automation and metering solutions. Test installations were performed on an Intel Core i3-5005U 2GHz personal computer with 8 GB RAM, running Ubuntu 22.10 64-bit in Virtual Box on a Windows 8.1 host system. PostgreSQL 14.5, MongoDB 6.0.2, and InfluxDB 1.8.10 were installed directly in the Ubuntu guest system, following the base installation method described in the technical documentation [4, 5, 6].

In the subsequent stage, a test program was developed using NodeRED to read, send, and write sensor data to the investigated DBMSs. The program utilized building blocks provided by NodeRED to write up to 1 million readings to each database. Data was collected in an accommodation room from February 1st to February 28th, 2023, with a sampling time of 1 second. Four measurements were recorded: temperature, humidity, CO2 concentration, and VOC levels, each stored as a 64-bit float variable. The sample indoor air quality (IAQ) sensor, comprising a DHT21 [7] and SGP30 [8], was connected to the Orange Pi® [9] SBC and sent the data over the local network to each DBMS using the standard line protocol (see Fig. 2).

Table 1: Functionality comparison of the selected DBMS.

	PostgreSQL	MongoDB	InfluxDB
KNX	No[1]	Yes	Yes[2]
LCN	No[3]	No	Yes[2]
MODBUS TCP	No	No	Yes[2]
MQTT	No[4]	No[4]	Yes[2]
MBus	No	No	No
PC	Yes	Yes	Yes
SBC	Yes	Yes	Yes
Cloud	Yes[5]	Yes[5]	Yes[5]
C#	Yes	Yes	Yes
Python	Yes	Yes	Yes
Java	Yes	Yes	Yes
C++	Yes	Yes	Yes

[1] Possibly, using additional software and *pgknx* add-on for address handling.
[2] Using Telegraf[25], data collection agent for InfluxDB.
[3] Available, with dedicated software parts, as presented in [17].
[4] Not directly, but brokers (e.g. *mosquitto*) and solutions like NodeRED offer this option.
[5] Available pre-built docker images and interoperability with all leading cloud solutions (e.g. *Kubernetes*) and cloud providers (*AWS®*, *Azure®*, *GoogleCloud®*).

Results

Analysis of functionality

A database for storing and further processing of measurement data should make it possible to collect data from existing as well as newly installed sensors, also creating an integration point for various systems. Thus, relying on the technical documentation and inspection of configuration files, the following features was considered:

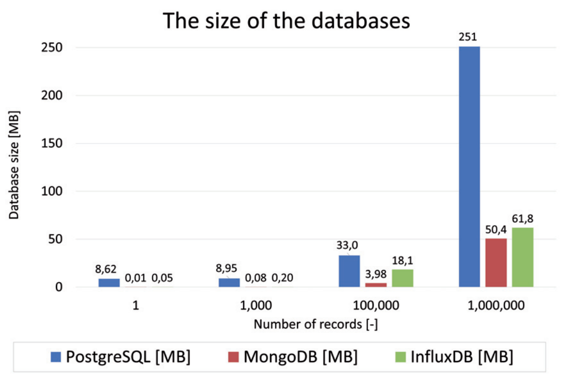
- a possibility of integration with two popular building automation systems (KNX, LCN),
- receiving of data using the most popular protocols used in automation (Modbus, MQTT, MBus),
- installation methods and the possibility of remote integration with the cloud,
- the availability of data reading interfaces, including the availability of APIs in the most popular programming languages, which can enable the development of original algorithms to analyze data and energy management in building.

Modbus protocol has been adopted both in traditional industrial automation and building solutions (e.g. for control of HVAC equipment), MQTT is becoming popular, especially in the field of so-called *smart-gadgets* (smart-sockets, air-quality sensors, vacuum robots, etc.). MBus is now being made available to consumers [24], including in Poland together with the installation of smart metering, so its handling may be important in terms of using data from smart-meters and raising end-user awareness about energy consumption. In the case of building systems and studied protocols, it is assumed that a suitable network gateway.

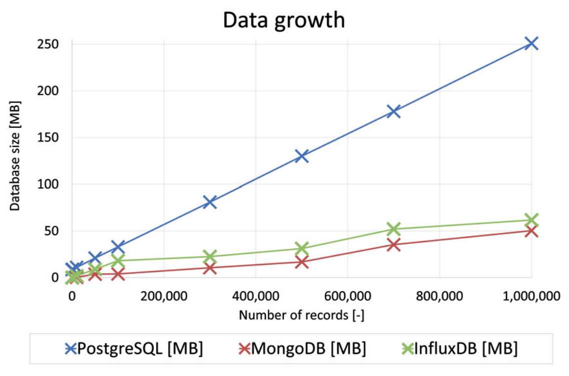
Considering the installation method, the possibility of installation on single PCs and SBC such as Raspberry PI, but also the possibility of using containerization and cloud installation was inspected. The results was summarized in Table 1.

Table 2: Summary of the response times

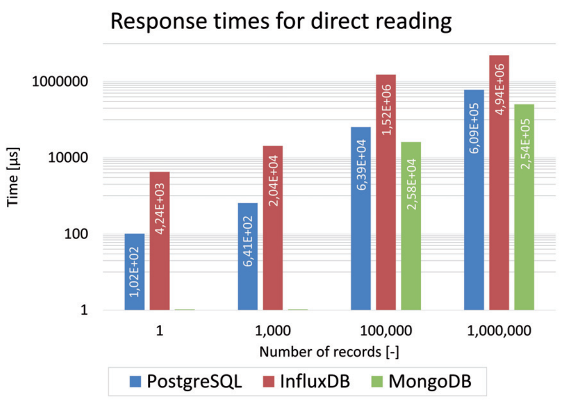
	Direct reading [ms]				Grafana [ms]			
Number of records	1	1,000	100,000	1,000,000	1	1,000	100,000	1,000,000
PostgreSQL	0.100	0.640	63.94	608.8	114.4	173.0	339.4	3694
InfluxDB	4.240	20.40	1519	4939	90.60	125.2	577.2	3982
MongoDB	<1	<1	25.80	253.6	0.761	6.630	563.2	5448



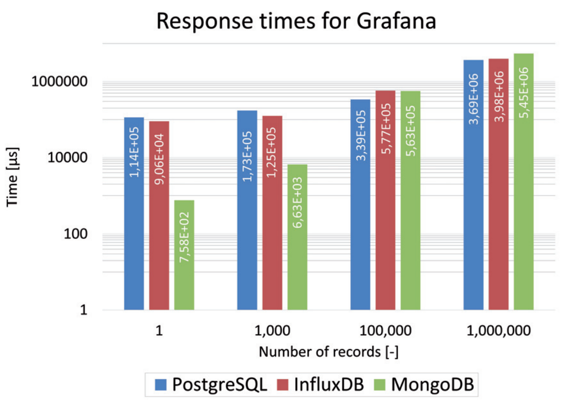
(a) Comparison of disk space consumption



(b) Growth of consumed disk space as a function of the number of records



(c) Comparison of response times for direct reading.



(d) Comparison of query times for reading through Grafana.

Fig. 3: Final overview for the analyzed parameters.

The greatest possibilities in terms of integration with various protocols can be stated for InfluxDB, due to the availability of a dedicated data collection agent Telegraf [25], delivered as part of the DBMS and which can be installed together. It is worth noting that it can be used independently to communicate with the other two investigated databases. Regrettably, none of the analysed DBMS or their add-ons directly support the MBus protocol, but such extensions can be expected to become available in the nearest future.

The installation of each DBMS is supported on most popular operating systems (Windows®, macOS®, Linux), natively as well as through Docker containers, on SBC, personal computers, servers and in the cloud. For a selection of the popular programming languages (C#, Python, Java, C++), each provides an official package to connect to the discussed databases.

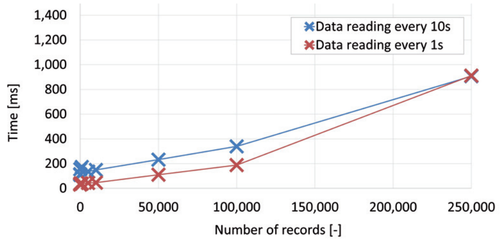
Disk space

The size of the bases on the disk for 1, 1,000, 100,000 and 1,000,000 records, respectively was measured. The comparison are shown in Fig. 3a.

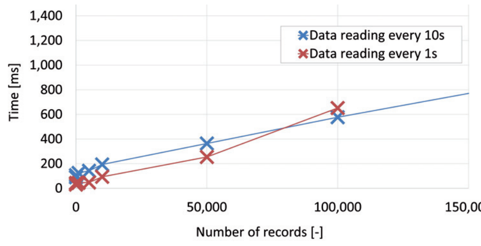
PostgreSQL achieved the least favourable results. Already for a single record the database reached a size on the order of 8 megabytes. The size of the MongoDB is the most satisfactory - it takes up the least disk space. The database volume of InfluxDB also reaches the optimal value, but with 100,000 records, the size was more than 4.5 times that of MongoDB. Reaching a very large number of records in the database, InfluxDB shows better data compression. With a million measurements, the size of the Influx database is only about 1.2 times larger than the Mongo database. The characteristics of the increase in the size of the databases as a function of the number of records was shown in Fig. 3b. For PostgreSQL the increment is pretty well linear, in the other two cases internal space optimization and compression mechanisms are presumably operating.

Query response times

Response times to queries for 1, 1,000, 100,000 and 1,000,000 records were also checked. The response time was measured when directly reading from the databases (Fig. 3c) and via the webclient Grafana (Fig. 3d). The total number of records collected and size of the database practically did not affect the response time, the biggest impact as



(a) PostgreSQL (Grafana)



(b) InfluxDB (Grafana)

Fig. 4: Changes in response time depending on the reading interval.

shown was the amount of data returned per query. In the case of Grafana, it was possible to re-run a query at a specified interval. During this tests when reducing the interval (e.g. from 10 to 1 second), a decrease in read times for InfluxDB and PostgreSQL was noticeable (see Fig. 4). MongoDB was the exception, as its response times did not significantly differ depending on the frequency of the call.

When reading directly from the database, the response time was checked for set of one type of data (SQL – column, TSDB – measurment), e.g. for temperature in defined time range. For the PostgreSQL database, the difference between the query for one type of data and the query for all data was about 2 times, and as more records were called, it increased to more than 3 times. In the case of an intermediary such as Grafana, the response time for the query every 10 seconds and every 1 second was similar. With 50 thousand records, the time exceeded 1 second.

With direct reading from InfluxDB, the time difference between the response for one data type and all data is noticeably greater than the difference for PostgreSQL. The difference does not increase proportionally to the number of records. Reading time for all data is about 10 times faster than reading for one type of data. Similar to PostgreSQL, when reading from Grafana, the response time for the query every 10 seconds and every 1 second was almost identical. With 25 thousand records, the time exceeded 1 second.

In MongoDB, it is not possible to show the response time in units smaller than milliseconds, so if the time is less, the value 0 is automatically returned. Only with 50,000 records, the database returned the response time value grather then 1 ms.

Reading data directly from databases using default client, MongoDB is characterized by the fastest access to data. PostgreSQL also achieved satisfactory results. InfluxDB fared less well. The results change dramatically if the reading is obtained using a external client, which in that case was Grafana. MongoDB communicates the least to the client and contrariwise as with direct reading it obtained the slowest access time for a larger number of records. PostgreSQL and InfluxDB got close performance. The numerical values of response times was shown in Table 2, and the averaged values for all studied variants, on a linear scale in Fig. 5.

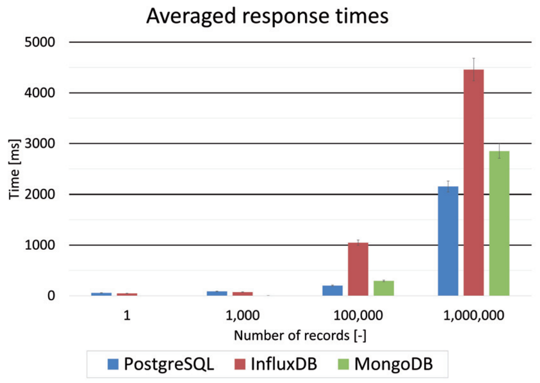


Fig. 5: Averaged final response times.

Discussion

PostgreSQL relational database is a heavily schema-based database, handling unique, often non-recurring data and strongly bound by relationships. The size of the database reaches the highest value compared to others.

Potentially better choice respecting the nature of the data in smart metering, would be to use the non-relational database MongoDB or the time-series-based InfluxDB. MongoDB is a document database with no specific schema, where each measurement is a separate document. It also has the advantage of ease of use and transparency. It is designed for high availability, with built-in data replication and automatic fragmentation. It takes up the least disk space compared to other databases.

Juxtaposing the two databases, MongoDB is definitely easier when searching for specific data from a specific time. However, it should be noted that the time series database definitely wins in terms of overall performance. Write throughput is definitely higher while using less disk space. The configuration of MongoDB is more complicated than in InfluxDB. The internal structure and data types must be chosen in advance. In a time series database, once the path to the database is established, the data is automatically saved [11, 26].

In general, InfluxDB is designed to handle large amounts of time series data efficiently, and most queries should execute in a matter of seconds or less [22]. But, for very large datasets or complex queries (e.g. retrieving the longer periods or using a math function in the query), it may take several seconds or even minutes for the query to complete, what was observed during the experiments.

Timings achieved by all tested databases (Fig. 5), in smart metering and smart-building purposes, due to the occurrence of large time constants, even with the direct use of retrieved data for the real-time control, are quite acceptable. When used for data visualization for the human operator, the delays will also be acceptable. In the case of cyclic readings, as was observed, the caching process and internal optimizations speeds up the subsequent data readings (Fig. 4).

Application guidelines

To recapitulate, the choice of a database is not clear-cut. In the case of smart metering, which involves a large amount of data measured over time, InfluxDB appears to be the most suitable option.

Data from sensors and meters inherently follow a time series format, and processing them is more intuitive for operators [27]. InfluxDB stands out as an advantageous choice because it is specifically designed for time series data and provides tools like Telegraf [25] to facilitate integration with multiple data sources. It should be sufficient for typical applications. Additionally, due to the structured nature of the data using timestamps and tags, the incorporation of new sensors and data sources does not necessitate changes to the overall database structure.

MongoDB differs the most in terms of the structure of the data, where it is stored as a set of JSON documents, it was also characterized by the fastest read time for direct queries and the smallest growth in disk space. So it is suitable, as the manufacturer recommends, for big-data applications. In smart building applications, the way data is stored can benefit from an event-driven approach. However, the setup and operation can be less intuitive, especially for staff familiar with the SQL-like environment.

PostgreSQL has the advantage of using the industry-standard SQL language and the knowledge of relational databases by a large number of technical staff. Problematically, before data collection begins, the structure of the tables in which the data would be stored must be designed; if this is done incorrectly, future changes or extension with new data sources can be demanding.

The use of one database does not prohibit the simultaneous use of others. PostgreSQL is frequently employed in individual devices or software across various systems, such as access control. InfluxDB will be suitable for use as the main database integrating data from multiple systems within a particular building or group of them (campus), allowing easy communication with multiple standards and data visualization. MongoDB would appear to be a good candidate for complex analytical systems with dedicated software, aggregating a lot of data, with different structures, coming from different sources and using, for example, machine learning.

In closing, we can mention that, InfluxDB, among other factors, based on the background presented in this paper, has been deployed in KAE PŁ* as open source solution for collecting data from multiple BACS systems, interoperating with Grafana [18], Telegraf [25] and gateways shown in Fig. 6.

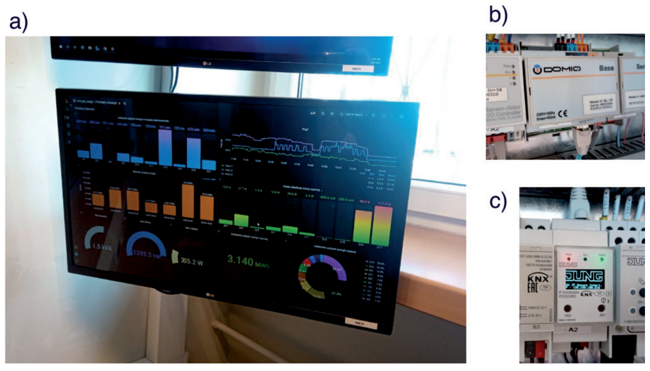


Fig. 6: Real application of InfluxDB working with the Grafana visualization tool (a), retrieving data from the BACS network gateways (b, c) via Telegraf.

Authors:
Eng. Sandra Włostowska, e-mail: 216219@edu.p.lodz.pl
Eng. Julia Szabela, e-mail: 216207@edu.p.lodz.pl
M.Sc. Adrian Chojecki, e-mail: adrian.chojecki@p.lodz.pl;
Prof. Piotr Borkowski, e-mail: piotr.borkowski@p.lodz.pl;
**Katedra Aparatów Elektrycznych, Politechnika Łódzka ul. Stefanowskiego 18/22, 90-924 Łódź, Poland.*

REFERENCES

[1] Elmasri R., Navathe S.: Wprowadzenie do systemów baz danych, Wydawnictwo Helion, Gliwice, 2005.
[2] Martin J.: Managing the Data-base Environment Prentice-Hall, Englewood Cliffs, , NJ, 1983.
[3] Haerder T., Reuter A.: Principles of Transaction-Oriented Database Recovery, ACM Comput. Surv., pp.287–317, Dec. 1983.
[4] PostgreSQL [web page] www.postgresql.org/, [Accessed on 9 Apr. 2023.].
[5] MongoDB, [web page] www.mongodb.com/. [Accessed on 9 Apr. 2023.].
[6] InfluxDB, [web page] www.influxdata.com/, [Accessed on 9 Apr. 2023.].
[7] DHT 21 | Temperature and humidity module, [web page] cdn-learn.adafruit.com/downloads/pdf/ dht.pdf [Accessed on 23 Mar. 2023.].
[8] SGP30 | Sensirion CO2 Sensor | Datasheet, [web page] sensirion.com/products/catalog/SGP30 [Accessed on 23 Mar. 2023.].
[9] Orange Pi Zero | Single Board Computer, [web page] www.orangepi.org [Accessed on 23 Mar. 2023.].
[10] Łacheciński S.: Składowanie i przetwarzanie danych temporalnych w świetle wymagań standardu języka SQL ISO/IEC 9075 Przeglad Elektrotechniczny. 10/2020. 184-191.,
[11] Geetha A., Jamuna K.: ICICES : International Conference on Information Communication and Embedded Systems, Smart metering system, Feb. 2013.
[12] Olivares-Rojas J. C., Reyes-Archundia E., Gutiérrez-Gnecchi J. A., Molina-Moreno I.,Cerda-Jacobo J., Méndez-Patiño A. A Comparative Assessment of Embedded Databases for Smart Metering Systems, IEEE PES Innovative Smart Grid Technologies Conference-Latin America, Sept. 2021
[13] Directive 2010/31/EU of the European Parliament and of the Council of 19 May 2010 on the energy performance of buildings (recast), Dec. 2018,
[14] Komisja PE za nowymi standardami efektywności energetycznej budynków, [web page] www.stooq.com/n/?f=1535810&c=0&p=4/, [Accessed on 23 Feb. 2023.].
[15] Bader A., Kopp O., Falkenthal M. Survey and comparison of open source Time Series Databases. Datenbanksysteme für Business, Technologie und Web, 2017.
[16] Grzesik P., Mrozek D. Comparative analysis of time series databases in the context of edge computing for low power sensor networks, ICCS 2020: 20th International Conference, Amsterdam, Springer, Proceedings, Part V 20, pp. 371-383, 2020.
[17] Ambroziak A., Borkowski P., Sienicki A.: IEICE Technical Report; IEICE Tech. Rep. OPC server for DOMIQ building automation system, pp. 165–169, 2014.
[18] Grafana, [web page] www.grafana.com/, [Accessed on 23 Mar. 2023.].
[19] pgAdmin Reviews and features, [web page] www.capterra.com/p/210772/pgAdmin/ reviews/, [Accessed on 10 Apr. 2023.].
[20] DB-Engines Ranking, [web page] db-engines.com/en/ranking/ [Accessed on 20 Apr. 2023.].
[21] Sadalage P., Fowler M.: Addison-Wesley, NoSQL distilled: a brief guide to the emerging world of polyglot persistence, 2013.
[22] Last M., Kandel A., Bunke H.: World Scientific Data Mining in Time Series Databases, 2004.
[23] McBride, Dave B.: Survey of time series database technology, Mar. 2020., Survey of time series database technology. Wallingford, UK Centre for Ecology & Hydrology, 33pp.
[24] Protokół MBus w licznikach Tauron, [web page] https://amiplus.tauron-dystrybucja.pl/ o-amiplus/ [Accessed on 23 Mar. 2023.].
[25] Telegraf [web page] https://www.influxdata.com/ telegraf/ [Accessed on 23 Mar. 2023.].
[26] Balis B., Bubak M., Harezlak D., Nowakowski P., Pawlik M., Wilk B, Towards an operational database for real-time environmental monitoring and early warning systems Procedia Computer Science, Vol. 108, pp. 2250-2259, 2017.
[27] Riaz Z., Parn E., Edwards D., Arslan M., Shen Ch., Pena-Mora F.,: BIM and sensor-based data management system Journal of Engineering, Design and Technology. 15. pp. 738-753, 2017.