BD4NRG QUERY ENGINE – INTUITIVE, EFFICIENT AND FEDERATED QUERYING ON BIG DATA

Konstantinos Touloumis, Evangelos Karakolis, Panagiotis Kapsalis, Vangelis Marinakis and John Psarras
Decision Support Systems Laboratory, School of Electrical and Computer Engineering, National Technical University of Athens, 9 Iroon Polytechniou Str., 15773 Athens, Greece

ABSTRACT

Due to the rapid development of Internet of Things (IoT) technology during the last decade, there has been recorded a significant growth in the size of data collected by data warehouses, especially on the ones connected to sensors and meters on the energy sector. Querying such data, gathered by heterogeneous data sources is a complicated task in terms of space allocation, scalability, and integration. Big data analytics methods try to address the problem of querying big data by bringing together different data sources of different formats with a high level of abstraction. This paper presents a query engine that, on the one hand addresses the problem of querying big data stored on different and heterogeneous databases and on the other hand, simplifies the steps needed by the user for defining and executing queries on the aforementioned databases through an intuitive and easy to use interface, so as to facilitate high level analytical services. Particular attention is given to securing the tool by employing access resource management thus preventing unauthorized users from accessing the requested resources.

KEYWORDS

Big Data Querying, Big Data Analytics, Energy Sector, Federated Querying

1. INTRODUCTION

With the exponential growth of networking capabilities, there has been noticed a rapid growth in the volume of data collected in databases, gathering data from various sources on the energy sector such as energy consumption data (Mohammed et al., 2019). Traditional databases fail to scale up to the huge volumes of data collected, thus, making querying a complex and inefficient task (Nadikattu, 2020). Data warehouses try to bridge the gap by using multidimensional models to effectively represent the data they store. Still, with the evolution of IoT they fail to respond to the massive volumes of collected data on the energy sector, thus introducing difficulties in efficient storing, preprocessing, and querying (Marinakis & Doukas, 2018). Data lakes have been proposed as they maintain heterogenous data sets in their original format. They aim at providing scalable and flexible knowledge discovery and querying. Moreover, they reduce the cost of storing and integrating data. However, serious complexity is introduced during query execution (Endris et al., 2019).

In this context, big data analytics methods allow efficient querying of heterogeneous data sources. According to (Zhang et al., 2017) big data is characterized by its variety, velocity, volume, value and veracity (5V's of big data). Variety indicates that data comes from different sources and can be structured, semi-structured and unstructured. Volume indicates that the size of data is growing rapidly and is not easily manageable. Velocity indicates that big data can be used as streams to maximize its value. Veracity indicates that not all data on such high volume is going to be clean, hence the accuracy of data analysis is going to be affected. Value indicates that acquired data should be useful for knowledge mining and providing user related services, as investing in big data technologies comes at a high cost for interested companies.

One sector that can be highly benefited from big data analytics technologies is the energy sector. This is because recent developments in the energy domain such as smart grids and buildings result to the generation of vast amounts of data, that cannot be easily analyzed without big data analytics technologies and techniques. Of course, several energy companies are already benefitted from big data analytics, however there is still significant potential for novel services based on big data analytics and artificial intelligence (AI) and for

improving already developed use cases (Chui et al., n.d.). To this end, there is a variety of projects that aim to unlock the full potential of big data analytics for the energy domain. This is done by private businesses internally, as well as globally by facilitating data sharing combining different data sources, instead of siloed approaches that are followed within a single organization. For instance, the I-NERGY project (Karakolis, et al., 2022) aims to unlock the full potential of AI on the energy sector through innovative services such as anomaly detection in smart buildings (Karakolis, et al., 2022), as well as electrical load forecasting (Pelekis et al., 2022), that cover the full energy value chain. Moreover, the MATRYCS project (MATRYCS, 2022) provides holistic energy services in smart buildings using big data. Also, BD4NRG project (BD4NRG, 2022) aims to establish new market opportunities in the energy sector using big data.

Such projects are huge and complicated, providing different and independent services on top of different and heterogeneous datasets gathered from different data sources and technologies. To derive value from these datasets, efficient querying technologies are required one the one hand, and on the other hand, combination of different and heterogeneous datasets should be made feasible. Moreover, querying is mostly related to programmers and data scientists, however energy domain experts can pose more relevant energy related queries as they are more familiar with this domain. In this context, an intuitive interface for creating and executing queries against the provided databases, without writing any SQL code is highly appreciated by domain experts. To deal with the huge volumes of data stored in data warehouses and assist data analysts and customers to extract useful and meaningful data, several query engine services and applications have been developed. However, they pose several limitations including ones related to compatibility with different technologies, security and access control management, efficiency and performance.

In this context, the research at hand presents the BD4NRG Query Engine application. This component is responsible for allowing its users explore, combine and express complex queries without any prior knowledge of any programming language, as well as executing these queries in an efficient manner, facilitating the execution of complex queries that combine different datasets from different and heterogeneous data sources and technologies including data streams alongside acceptable response time. This component can also be used by other applications through APIs as well as through direct connections to the main querying technology, to facilitate several data analytics applications with the requested data. It is worth mentioning that on top of these functionalities a security and access control management component has been employed to make sure that users have access only to data that are authorised to.

2. BACKGROUND AND RELATED WORK

According to (Mohammed et al., 2019) the challenges on big data querying are the following: Streaming of large-scale data from IoT, as the huge volumes of data during data acquisition phase makes it difficult to decide which data must be discarded and which to maintain. The data heterogeneity, as most data sources do not follow a structured format, thus, transforming a data source of unstructured format to structured is a major challenge. Pre-processing, cleaning and analyzing data is a complex task both due to scalability and complexity of data to be analyzed. Also, data storage and space allocation should be optimized. Security and privacy concerns are becoming extremely important for large scale integration and deployment of the IoT. Finally, visualizing querying results is very challenging as domain experts should be able to easily extract useful knowledge in a straightforward manner (Agrawal et al., 2015). This paper focuses mostly on big data querying.

Of course, a big data querying component should comply to certain functional and non-functional requirements as recognized in literature (Bohlouli et al., 2013). Some of the functional requirements include:

Integration: the query engine must be capable of being connected and fetching data from different data sources. This is because complex big data systems process data from different data sources that are not always stored in the same storage or database. There is a wide variety of SQL and NoSQL database technologies and each one focuses on different aspects, including general purpose big data databases focusing on performance (e.g. MongoDB (MongoDB, 2022)), timeseries databases (e.g. InfluxDB (influxdata, 2022)) for efficient timeseries operations, and conventional SQL ones for typical Online Transaction Processing (OLTP), as well as data lakes.

Statistical analysis: the query engine must be capable of performing simple and complex analysis like calculating statistics on aggregated data and providing Online Analytical Processing (OLAP) capabilities.

Exploration: the query engine should provide visual analytics for interacting with data to extract useful knowledge.

Decision support: the query engine should provide mechanisms for assisting decision making of stakeholders on both domain and non-domain decision problems.

Non-functional requirements include:

Scalability: the query engine should be able to handle large amounts of data.

Near real time monitoring: near real time monitoring is necessary especially for streaming data.

Resource efficiency: system resources should be utilized in an efficient way.

In this context, there are several big data querying technologies that address some of the presented functional and non-functional requirements. For instance, Trino (Trino, 2022) is a powerful distributed SQL query engine that facilitates big data analytics applications, offering low latency analytics, great scaling capabilities, query federation, simplicity (ANSI SQL compliant), as well as compatibility with the most well-known databases. Presto (Presto, 2022) offers similar functionalities, as it is fast, reliable, scalable and compatible the most well-known databases. Although Presto is a well-established solution in the market and Trino was built on top of it, the latter poses a few advantages over Presto. Specifically, it provides higher level of versatility when querying traditional data sources, non-relational databases and columnar databases. Moreover, it provides options to save resources. On the other hand, Presto provides better performance for processing specific file formats like ORC. It also, provides a library within Spark (Apache Spark, n.d.) executor (pandio, 2021). Since there are clear advantages in each technology, the selection of one option should be done according to the requirements of the specific use case. In this work, Trino was selected, as a large number of different database technologies is used, while the advantages of Presto are not relevant to the specific work.

In this context, several relevant query engine applications available in literature will be presented. Specifically, in (Linder et al., 2017) a big data system for collecting building data is presented. It depends on the notion of virtual objects. A virtual object is mapped to a real sensor through an ID by a specialized collector. A MySQL (Krogh, 2020) database is used to augment metadata with a virtual object and transmit it through Kafka (THEIN, 2014) messaging. Tokens are also used for securing data privacy. However, the research mentions that NoSQL (Atzeni et al., 2020) databases compromising between consistency, availability and partition tolerance and are not appropriate for acting at the storage level.

Similarly, in (Pau et al., 2022) and (Alexakis et al., 2022) another big data system for querying analytics on buildings sector is analyzed. The first step is data preprocessing to remove incomplete or inaccurate data. The transmission layer uses a Kafka broker for transmitting data. Kafka messaging offers high throughput and latency, but it is inefficient for time-based aggregation on data streams. Data is then harmonized and projected onto a common data model. A MongoDB service acts as the enriched data warehouse that stores the processed data. On top of this warehouse the Presto query engine is used for querying different data sources.

Besides SQL based query engines, several reasoning engines are presented in literature. They are based on semantic technologies to infer logical relations. Reasoning engine applications include the work of (Kapsalis et al., 2022) which presents a reasoning engine for smart building metadata management. The main technology that has been used is Neo4J graph database on top of RDF ontology data. Also (Anagnostopoulos et al., 2013) presented a reasoning engine for extracting qualitative temporal information using OWL language.

It is worth mentioning that a common problem that Big Data Analytics query engines (and projects) face is the identity and access control management on different data sources. Specifically, when there are many data providers, involved, alongside multiple other users from different organisations and levels of expertise, access to the available datasets should be restricted based on the provider's preferences. Hence, a user should have access only to resources he is authorised to. To this end, there are several technologies for identity and access control management such as FIWARE Keyrock (FIWARE, n.d.) and Keycloak (Keycloak, 2022), and several scientific publications addressing this aspect. For instance, (Kormpakis et al., 2022) present a framework for securing a visual analytics application for the energy sector, which enables only authorised access to energy related data, through Keycloak and role-based access control.

3. THE PROPOSED SOLUTION

In Figure 1 a high-level overview of the proposed big data architecture is presented focusing on the querying capabilities that BD4NRG Query Engine offers. It consists of several layers of functionalities, namely the data sources, the Data Storage, the Querying Component, the Querying and Analytics interfaces, alongside an Identity and Access Control Management layer and the main platform users.

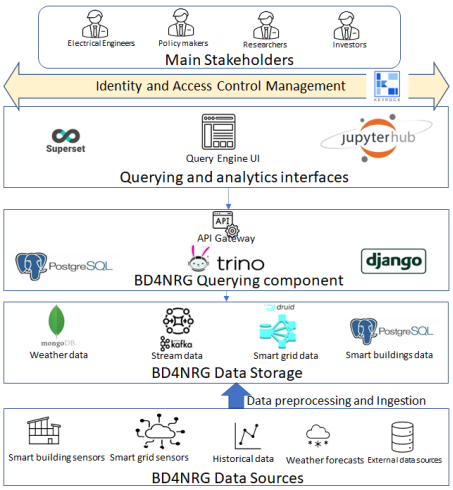

Figure 1. BD4NRG query engine high-level architecture

Specifically, the **BD4NRG data sources** include data from sensors and smart meters from both buildings and smart grids, weather related data including weather forecasts and actual weather measurements (e.g. temperature, humidity, wind speed), historical data of different kinds (e.g. fault incidents in smart grids) among others. All these data are pre-processed, curated, and harmonized to be stored to BD4NRG platform.

These data are stored to one of the databases of the **BD4NRG Data Storage layer**. Specifically, continuous flows of streaming data are ingested to BD4NRG platform through Kafka message broker, using protocols like MQTT and AMQP. Moreover, other types of data are stored to the databases into batches. Afterwards, all these data are stored to the database technologies employed for this component. In particular, weather data is stored to a MongoDB database, data from smart buildings are stored to a PostgreSQL database, while data from smart grids are stored to Apache Druid.

The **BD4NRG Querying component** is responsible for the execution of queries against the databases of the previous layer, and the combination of the available datasets. The main technology used in this component is Trino, a Big Data distributed Query Engine technology, which is an open-source, very efficient querying technology that can query exabytes of data and is used by some of the largest organizations in the world. Trino is able of connecting to all BD4NRG data sources, like the ones presented in the storage layer. Moreover, it allows query federation within a single query, and is compatible with the most well-known databases. Also, Trino enables queries to Kafka data streams, which is an important functionality for near real time analytics services. Also, it is ANSI SQL compliant and works with several well-known BI tools. It is worth mentioning that the BD4NRG Querying Component provides several APIs to execute queries on it. To this end, a Django web application has been developed to provide these APIs. Moreover, a PostgreSQL database has been employed to store dataset metadata. Several APIs of the Querying component are presented in Table 1.

Table 1. Querying component REST APIs

Description		HTTP Method	Endpoint URL	Params	Body
1.	Retrieve a list of the available datasets of the platform. For each dataset, some metadata are provided (e.g. title, a small description)	GET	/apis/get/datasources/	-	-
2.	Retrieve a list of all the available variables from all datasets of the platform.	GET	/apis/get/variables/	-	-
3.	Retrieve a list with all the variables that are included in a selected dataset. For each variable, some metadata are provided, such as its name, its dataset and its description.	GET	/apis/get/datasource/variables/	-	{ “catalog”: “druid”, “schema”: “druid”, “dataset”: “lsp01” }
4.	Add a new dataset to the list of available datasets.	POST	/api/get/{datasource_id}/datasource/	-	{ “dataset”: “lsp01” }

The **Querying and Analytics interfaces layer's** main goal is to assist users in exploring the available datasets, building, executing custom queries and performing exploratory analysis. This is done through several ways. First, an intuitive, user-friendly web interface has been developed to facilitate easy query creation and execution by non-IT-related users. To this end, the Query Engine UI communicates with the Querying component through APIs and directs users in defining the query step by step, selecting datasets, aggregations on the data tables, and the logical expressions that will be used for filtering the selected data. Furthermore, Apache Superset is employed to allow high level visualizations for all data sources connected to Trino. By using Superset, users are able of making visualizations for inferring statistics and further discovering relations between data. Last, JupyterHub has been employed to serve Jupyter notebooks to multiple users for facilitating the creation of custom services on the project assets. It is worth mentioning that many BD4NRG analytical services communicate with the BD4NRG Querying component through APIs to execute efficient and low latency federated queries.

The **identity and access control management component** aims to manage resource access within the platform. Keyrock IdM is used for monitoring user access, roles and permissions on trusted applications. Keyrock Idm is an open-source identity management tool developed by FIWARE, that can register store and query identity data, with the purpose of providing and validating access tokens to users to authenticate them to trusted applications. Wilma PEP Proxy is a policy enforcement proxy server that validates the received access tokens with Keyrock IdM. The latter uses the OAuth 2.0 protocol for authorizing access of client users to the resources owned by a host user without knowing the host's credential. By using this protocol an application can obtain limited access to a resource, on behalf of the resource owner by orchestrating an approval interaction. These components are used for securing the custom BD4NRG user interface, Apache Superset and JupyterHub.

The most important **users/stakeholders** of the platform include among others several EPES stakeholders (electrical engineers, data scientists TSOs, DSOs etc.), investors, researchers and policy makers.

4. INDICATIVE USE CASE

A Greek island has hybrid power station of RES storage and a network of smart meters installed on the grid (e.g. in smart buildings). All sensors and smart meters are connected to BD4NRG platform through IoT technologies. A PostgreSQL database named "sensor_values" is used to store data from different kind of smart meters. Each row is identified by the sensor's name and contains the average value of the sensor, the unit of measurement and the date of the measurement. A MongoDB named "sensor_weather" is used to store weather data including forecasts for wind speed, temperature, humidity and pressure along with the timestamp of the measurement. A building manager can use the web interface of BD4NRG Query Engine to query data to assess the average consumption of a building on different weather conditions. The course of action is the following:

The user defines datasets and variables he intends to query: he first selects the datasets to be queried and afterwards the variables he is interested to view. He can also define aggregation for each field (e.g. MAX, AVERAGE, COUNT). In this use case, he selects a smart building dataset located on a Greek island, and afterwards selects to view all sensor values and measurements over time. So, he selects the mean column, which shows the measurement value, the statistic_id column, which indicates what is the measurement of a specific record (e.g. the total energy consumption of a building), the unit of measurement and the date of measurement (upper left section of Figure 2).

The user combines the smart building data with weather data: he selects the weather dataset, along with the variables temperature, wind speed, air pressure and the time of the measurements which will be the field to join with the previous dataset and is common for both data sources (upper right section of Figure 2).

The user intends to view only weather data for the place in which the buildings are located: so, he should define the filters that will be applied on the selected fields. Many kinds of operators are supported, i.e he can select the temperature and define that it should be within a range. The user is also able to combine expressions using logical operators. Finally, he selects to filter by the place (bottom left section of Figure 2).

The user inspects the created query, executes it, views and downloads the results. Once he has finished with the query, he can view the generated SQL, and execute the query. After a couple of seconds, he views the results which are downloadable to a csv file (bottom right section of Figure **2**).

Of course, the Query Engine user interface enables many more functionalities such as data aggregations, group by and having statements, multiple joins and filters, limit, ordering and more. However, they will not be presented in the context of this publication. Superset and JupyterHub will not be presented either.

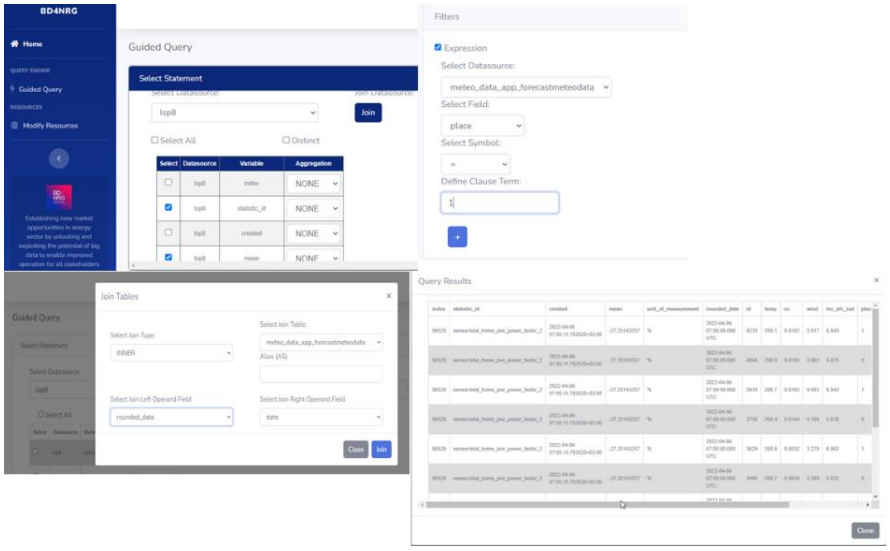

Figure 2. BD4NRG query engine web interface

5. DISCUSSION

BD4NRG Query Engine is a state-of-the-art big data query engine that fulfills all the functional and non-functional requirements of a big data analytics querying component, as presented in section 2. Specifically, in terms of functional requirements, it fulfills the requirement for **integration**, as it is capable of connecting and retrieving data from different and heterogeneous data sources. Moreover, it enables **statistical analysis** (second functional requirement) since it facilitates the creation and efficient execution of aggregation queries and can be used for OLAP. Furthermore, it provides functionalities for **exploration** (3rd functional requirement) through its intuitive user interface, as well as through APIs and a visual analytics service such as Apache Superset, which facilitates further the data exploration though data visualisations and dashboards. Last but not least, the proposed technical solution assists **decision support** (4th functional requirement) through its intuitive user interface, as it helps users build and save their own queries and results, as well as through Apache Superset and its data visualisation, dashboard and reporting capabilities, and also through JupyterHub, which enables users create their own custom services, using the BD4NRG Query Engine.

In terms of non-functional requirements, the proposed solution is **highly scalable** (1st non-functional requirement), as the main technology, that has been selected, is used by some of the largest organisations globally, and can query exabyte scale data lakes. Regarding the requirement for **near real-time monitoring**, the BD4NRG query engine enables not only near real-time monitoring but also near real-time querying, through queries to Kafka streams of data. Last, the proposed query engine is **efficient in terms of resources** (3rd requirement) as Trino is a highly parallel and distributed querying technology that facilitates efficient, low latency analytics queries.

Of course, the proposed query engine goes beyond the aforementioned requirements, as it facilitates also queries to Kafka streams in near real-time data and query federation between different and heterogeneous data sources and technologies. Moreover, it facilitates easy query creation and execution by stakeholders that are not experienced with programming and SQL. Beyond the user interface, the proposed solution provides several capabilities for data visualisations and custom services creation by users. Last but not least, usage of BD4NRG Query Engine is protected through an Access Control and Identity Management service, that secures that a user can view only the resources for which he has permission to view.

Compared to other query engine applications, the proposed one not only focuses on efficient querying of the available resources, but also on the accompanying services for maximizing user profit by providing also an intuitive, user-friendly interface, visual analytics, and custom service creation capabilities. Moreover, it provides security and access control, as these attributes are extremely important on any big data platform. Last but not least, it enables near real-time queries to data streams.

6. CONCLUSSION AND FUTURE OUTLOOK

In the publication at hand, BD4NRG Query Engine application for big data analytics querying to energy related big data was presented. It is a big data querying service that fulfills the requirements of a state-of-the-art big data query engine as presented in literature and constitutes an end-to-end solution on big data querying, providing also visual analytics and custom service creation capabilities as well as an intuitive web interface.

Regarding future extensions of this publication, they include the validation of the presented application on a production environment in terms of user experience, security and stress tests. To this end, also multi-node installations of the querying technologies will be examined. Future work will also employ metrics for benchmarking the time complexity for executing queries, along with the space utilization, to compare the proposed framework against similar approaches using alternative query engines. Moreover, the proposed solution is envisaged to be extended to support also queries against data sources that do not support SQL queries, such as graph databases and ontologies. Moreover, compatibility with IDSA data spaces will be examined to enable efficient and sovereign data sharing among different organisations (Á. Alonso et al., 2018).

ACKNOWLEDGEMENT

This research has been co-funded by European Union's Horizon 2020 research and innovation programme under grant agreement no. 872613.

REFERENCES

Agrawal, R., Kadadi, A., Dai, X., & Andres, F. (2015). Challenges and opportunities with big data visualization. *7th International ACM Conference on Management of Computational and CollEctive Intelligence in Digital EcoSystems, MEDES 2015*. https://doi.org/10.1145/2857218.2857256

Alexakis, K., Kapsalis, P., Mylona, Z., Kormpakis, G., Karakolis, E., Ntanos, C., & Askounis, D. (2022). Intelligent Querying for Implementing Building Aggregation Pipelines. *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–6. https://doi.org/10.1109/IISA56318.2022.9904384

Alonso, Á., Pozo, A., Cantera, J., de la Vega, F., & Hierro, J. (2018). Industrial Data Space Architecture Implementation Using FIWARE. *Sensors*, *18*(7), 2226. https://doi.org/10.3390/s18072226

Anagnostopoulos, E., Batsakis, S., & Petrakis, E. G. M. (2013). CHRONOS: A Reasoning Engine for Qualitative Temporal Information in OWL. *Procedia Computer Science*, *22*, 70–77. https://doi.org/10.1016/j.procs.2013.09.082

Apache Spark. (n.d.). Unified engine for large-scale data analytics. *Https://Spark.Apache.Org/*.

Atzeni, P., Bugiotti, F., Cabibbo, L., & Torlone, R. (2020). Data modeling in the NoSQL world. *Computer Standards and Interfaces*, *67*. https://doi.org/10.1016/j.csi.2016.10.003

BD4NRG. (2022). *https://www.bd4nrg.eu/*. Https://Www.Bd4nrg.Eu/.

Bohlouli, M., Schulz, F., Angelis, L., Pahor, D., Brandic, I., Atlan, D., & Tate, R. (2013). Towards an integrated platform for big data analysis. In *Integration of Practice-Oriented Knowledge Technology: Trends and Prospectives* (Vol. 9783642344718). https://doi.org/10.1007/978-3-642-34471-8_4

Chui, M., Collins, M., & Patel, M. (n.d.). *IoT value set to accelerate through 2030: Where and how to capture it*. Https://Www.Mckinsey.Com/Capabilities/Mckinsey-Digital/Our-Insights/Iot-Value-Set-to-Accelerate-through-2030-Where-and-How-to-Capture-It.

Endris, K. M., Rohde, P. D., Vidal, M. E., & Auer, S. (2019). Ontario: Federated Query Processing Against a Semantic Data Lake. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), 11706 LNCS*. https://doi.org/10.1007/978-3-030-27615-7_29

FIWARE. (n.d.). *Identity Manager - Keyrock*. Https://Fiware-Idm.Readthedocs.Io/En/Latest/.

influxdata. (2022). *InfluxDB Time Series Data Platform*. Https://Www.Influxdata.Com/.

Kapsalis, P., Kormpakis, G., Alexakis, K., & Askounis, D. (2022). Leveraging Graph Analytics for Energy Efficiency Certificates. *Energies*, *15*(4), 1500. https://doi.org/10.3390/en15041500

Kapsalis, P., Kormpakis, G., Alexakis, K., Karakolis, E., Mouzakitis, S., & Askounis, D. (2022). A Reasoning Engine Architecture for Building Energy Metadata Management. *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–7. https://doi.org/10.1109/IISA56318.2022.9904419

Karakolis, E., Alexakis, K., Kapsalis, P., Mouzakitis, S., & Psarras, J. (2022). An end-to-end approach for scalable real time Anomaly detection in smart buildings. *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–7. https://doi.org/10.1109/IISA56318.2022.9904410

Karakolis, E., Pelekis, S., Mouzakitis, S., Markaki, O., Papapostolou, A., Korbakis, G., & Psarras, J. (2022). Artificial Intelligence for Next Generation Energy Services Across Europe - The I-NERGY Project. *IADIS International Conference E-Society 2022 (e-Society 2022)*.

Keycloak. (2022). *Keycloak*. Https://Www.Keycloak.Org/.

Kormpakis, G., Kapsalis, P., Alexakis, K., Pelekis, S., Karakolis, E., & Doukas, H. (2022). An Advanced Visualisation Engine with Role-Based Access Control for Building Energy Visual Analytics. *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–8. https://doi.org/10.1109/IISA56318.2022.9904353.

Krogh, J. W. (2020). MySQL Workbench. In *MySQL 8 Query Performance Tuning*. https://doi.org/10.1007/978-1-4842-5584-1_11

Linder, L., Vionnet, D., Bacher, J. P., & Hennebert, J. (2017). Big Building Data-a Big Data Platform for Smart Buildings. *Energy Procedia*, *122*. https://doi.org/10.1016/j.egypro.2017.07.354

Marinakis, V. (2020). Big data for energy management and energy-efficient buildings. *Energies*, *13*(7). https://doi.org/10.3390/en13071555

Marinakis, V., & Doukas, H. (2018). An advanced IoT-based system for intelligent energy management in buildings. *Sensors (Switzerland)*, *18*(2). https://doi.org/10.3390/s18020610

MATRYCS. (2022). *The MATRYCS Project*. Https://Matrycs.Eu/.

Mohammed, T. A., Ghareeb, A., Al-Bayaty, H., & Aljawarneh, S. (2019). Big data challenges and achievements: Applications on smart cities and energy sector. *ACM International Conference Proceeding Series*. https://doi.org/10.1145/3368691.3368717

MongoDB. (2022). *MongoDB*. Https://Www.Mongodb.Com/Home.

Motlagh, N. H., Mohammadrezaei, M., Hunt, J., & Zakeri, B. (2020). Internet of things (IoT) and the energy sector. In *Energies* (Vol. 13, Issue 2). https://doi.org/10.3390/en13020494

Nadikattu, R. R. (2020). Data Warehouse Architecture – Leading the next generation Data Science. *International Journal of Computer Trends and Technology*, *67*(9). https://doi.org/10.14445/22312803/ijctt-v67i9p113

pandio. (2021). What's the Difference Between Trino and PrestoDB? *Https://Pandio.Com/Difference-between-Trino-and-Prestodb/*.

Pau, M., Kapsalis, P., Pan, Z., Korbakis, G., Pellegrino, D., & Monti, A. (2022). MATRYCS—A Big Data Architecture for Advanced Services in the Building Domain. *Energies*, *15*(7), 2568. https://doi.org/10.3390/en15072568

Pelekis, S., Karakolis, E., Silva, F., Schoinas, V., Mouzakitis, S., Kormpakis, G., Amaro, N., & Psarras, J. (2022). In Search of Deep Learning Architectures for Load Forecasting: A Comparative Analysis and the Impact of the Covid-19 Pandemic on Model Performance. *2022 13th International Conference on Information, Intelligence, Systems & Applications (IISA)*, 1–8. https://doi.org/10.1109/IISA56318.2022.9904363

PostgreSQL. (2022). *PostgreSQL*. Https://Www.Postgresql.Org/.

Presto. (2022). *Presto: Free, Open-Source SQL Query Engine for any Data*. Https://Prestodb.Io/.

Thein, K. M. M. (2014). Apache Kafka: Next Generation Distributed Messaging System KHIN ME ME THEIN. *International Journal of Scientific Engineering and Technology Rescearch*, *3*(47).

Trino. (2022). *Trino | Distributed SQL query engine for big data*. Https://Trino.Io/.

Zhang, Y., Ren, J., Liu, J., Xu, C., Guo, H., & Liu, Y. (2017a). A Survey on Emerging Computing Paradigms for Big Data. *Chinese Journal of Electronics*, *26*(1), 1–12. https://doi.org/10.1049/cje.2016.11.016