УДК 004.62:004.75:004.65

M.V. TALAKH, V.V. DVORZHAK, Yu.O. USHENKO

TIME SERIES DATA MANAGEMENT IN SMART HOME SYSTEMS: BALANCING REAL-TIME ANALYTICS AND DATA STORAGE

Yuriy Fedkovych Chernivtsi National University, 2 Kotsjubynskyi Str. Chernivtsi, Ukraine, e-mail: m.talah@chnu.edu.ua

Анотація. Стаття представляє новий підхід до управління даними часових рядів у системах розумного будинку, зосереджуючись на балансі між аналітикою в реальному часі та ефективним зберіганням історичних даних. Гібридна архітектура, яка поєднує Azure CosmosDB для обробки даних у реальному часі та Azure Synapse для аналітики історичних даних, демонструє значні переваги в продуктивності. Система досягає прискорення до 165 разів у виконанні аналітичних запитів (від 3,3 секунди до 20 мілісекунд для запитів агрегації часових рядів) при одночасному зменшенні кількості операцій читання в 17 разів. Впровадження оптимізаційних стратегій, таких як запис на основі зміни стану та інтервальне зберігання, значно зменшує обсяг даних при збереженні темпоральної цілісності даних. Система демонструє лінійну масштабованість, здатність обробляти до 1 мільйона операцій запису на секунду та досягає коефіцієнтів стиснення до 5:1 у CosmosDB та 10:1 у Synapse.

Ключові слова: дані часових рядів; системи розумного дому; Інтернет речей; аналітика в режимі реального часу; зберігання історичних даних; гібридна архітектура бази даних; оптимізація даних; NoSQL; реляційні бази даних; управління великими даними.

Abstract. The article presents a novel approach to managing time-series data in smart home systems that balances real-time analytics with efficient historical data storage. A hybrid architecture combining Azure CosmosDB for real-time data processing and Azure Synapse for historical data analytics demonstrates significant performance advantages. The system achieves up to 165-fold acceleration in analytical query execution (from 3.3 seconds to 20 milliseconds for time-series aggregation queries) while reducing the number of read operations by 17. Implementing optimization strategies, such as state-based recording and interval-based storage, significantly reduces data volume while maintaining temporal data integrity. The system demonstrates linear scalability, handling up to 1 million write operations per second, and achieves compression ratios of up to 5:1 in CosmosDB and 10:1 in Synapse.

Keywords: time series data; smart home systems; IoT; real-time analytics; historical data storage; hybrid database architecture; data optimization; NoSQL; relational databases; big data management.

DOI: 10.31649/1681-7893-2025-50-2-54-61

INTRODUCTION

The rapid proliferation of Internet of Things (IoT) devices in smart home systems has led to an exponential increase in the generation of time series data [1-3]. With the number of connected IoT devices expected to reach billions in the coming years, managing the massive volumes of time-stamped sensor data has become a critical challenge for both researchers and practitioners in the field of smart home technology.

Smart home data exhibits a dual nature that presents unique challenges for data management systems:

a) Real-time updates - sensors continuously generate data that requires immediate monitoring and alerts processing [4];

b) Historical trends – long-term storage of historical data is essential for identifying patterns, performing predictive analytics, and optimizing home systems over time [2, 5].

The challenge lies in efficiently storing and processing this data while simultaneously providing real-time analytics and comprehensive historical analysis [2, 6]. Traditional data management approaches, which typically rely on either transactional databases for real-time operations or data warehouses for analytical queries, often struggle to meet both requirements efficiently for time-series data generated by IoT devices.

Current research in this field has focused on various aspects of IoT data management, including real-time data analytics [1], time series data generation in IoT [2], and smart home energy management [4]. However, there remains a gap in comprehensive solutions that effectively balance the competing demands of real-time processing and historical analysis for time series data in smart home environments.

To address these challenges, this paper presents a novel hybrid architecture that combines NoSQL and relational database technologies [6]. This approach leverages the strengths of both paradigms: the high-throughput, low-latency capabilities of NoSQL databases (specifically Azure CosmosDB) for real-time time series data ingestion and querying, coupled with the robust analytical processing power of relational data warehouses (Azure Synapse) for complex historical time series analysis.

The objectives of this study are to:

1. Develop a hybrid architecture that effectively manages both real-time and historical time series data from smart home environments.
2. Implement and evaluate time series data optimization strategies for efficient storage and retrieval, focusing on minimizing data redundancy and maximizing query performance.
3. Assess the performance and scalability of the proposed system in handling large-scale time series data, comparing it to traditional methods in terms of storage efficiency, retrieval speed, and computational resource utilization.
4. Demonstrate the practical application of the system through a case study on time series data processing for temperature monitoring in a smart home.

By addressing the challenges of high-dimensional time series data and the need for long-term dependency modeling [2], our proposed solution aims to significantly contribute to the field of smart home data management, providing a scalable and efficient framework for handling the ever-increasing volumes of IoT-generated time series data.

The primary contributions of this work are threefold. First, we present a comprehensive hybrid architecture explicitly optimized for smart home time-series data, demonstrating that the strategic integration of NoSQL and analytical warehouse technologies delivers superior performance compared to monolithic approaches. Second, we introduce time-series-specific optimization strategies that achieve significant storage reductions while preserving temporal data integrity. Third, through systematic experimentation, we quantify performance trade-offs in hybrid architectures, providing empirical evidence for architectural decision-making in IoT systems [9,10].

1. THEORETICAL AND METHODOLOGICAL FOUNDATIONS

1.1. Characteristics of IoT Data in Smart Home Systems

Time series data in IoT systems is characterized by its temporal nature, high volume, and potential for irregularity [7]. Each data point is typically associated with a timestamp and represents a measurement or state at that time. In smart home environments, this data comes from various sensors monitoring temperature, humidity, motion, energy consumption, and other parameters.

Key characteristics of smart home data include: a) High-frequency data generation - sensors continuously produce data at varying intervals, ranging from seconds to hours [2]; b) Data type diversity - smart homes incorporate a wide array of sensors, each generating different types of data [2, 3]; c) Mixed time series types - smart home data consists of both discrete and continuous time series [3]; d) Irregular data arrival patterns - some sensors generate constant streams of data, while others only transmit information when a state change occurs [4]; e) Temporal importance - each data point must be accurately timestamped to enable proper analysis of trends and patterns over time [1, 2].

1.2. Challenges in Storing and Processing Time Series Data

The management of time series data in smart home systems presents several challenges:

a) High data volume: Continuous data generation leads to rapid accumulation of large datasets [8].

b) Real-time processing: Smart home systems often require immediate processing of incoming data [1].

c) Long-term storage and efficient retrieval: Smart home systems generate vast amounts of historical data that need to be stored efficiently and remain easily accessible for extended periods [11]. This presents challenges in balancing storage costs with query performance.

d) Data heterogeneity: The diverse nature of smart home data complicates data modeling and storage strategies [9].

e) Scalability: Data management systems must scale horizontally to accommodate increasing data volumes [5].

f) Query performance: Efficient execution of both simple point queries and complex analytical queries is crucial [12].

While our proposed architecture addresses most of these challenges comprehensively, some aspects, such as data heterogeneity, are partially addressed and may require further investigation in future research.

1.3. Comparison with Specialized Time Series Databases

Specialized time-series databases such as InfluxDB and TimescaleDB offer purpose-built solutions for time-series data management. InfluxDB provides high write throughput and efficient compression for time-stamped data, while TimescaleDB extends PostgreSQL with time-series optimizations, including automatic partitioning and continuous aggregates. However, these specialized solutions often struggle to provide comprehensive analytical capabilities comparable to modern cloud-based data warehouses, particularly for complex multidimensional queries and long-term historical analysis, which are essential in smart home environments.

Our hybrid approach addresses these limitations by separating concerns: NoSQL handles high-velocity ingestion (on par with specialized TSDBs) while the data warehouse provides robust analytical capabilities (superior to pure TSDB solutions). This architecture achieves both the write throughput of specialized time series databases and the analytical flexibility of data warehouses, avoiding the performance-analytics trade-off inherent in single-database solutions [10, 11].

2. METHODOLOGY

To address these challenges, we propose a hybrid architecture combining NoSQL and relational database technologies. The methodology consists of the following components.

2.1. System Architecture

Our proposed solution employs a hybrid architecture that strategically separates time series data handling into two specialized storage tiers based on data recency and access patterns. This architecture addresses the dual requirements of smart home systems: immediate access to recent sensor readings for real-time monitoring and efficient long-term storage for historical trend analysis.

Figure 1 illustrates the overall system architecture, showing the complete data flow from IoT sensors through the real-time processing layer to the historical analytics layer. The architecture clearly separates hot storage (CosmosDB) for recent, high-frequency access and cold storage (Synapse) for analytical queries on historical data.

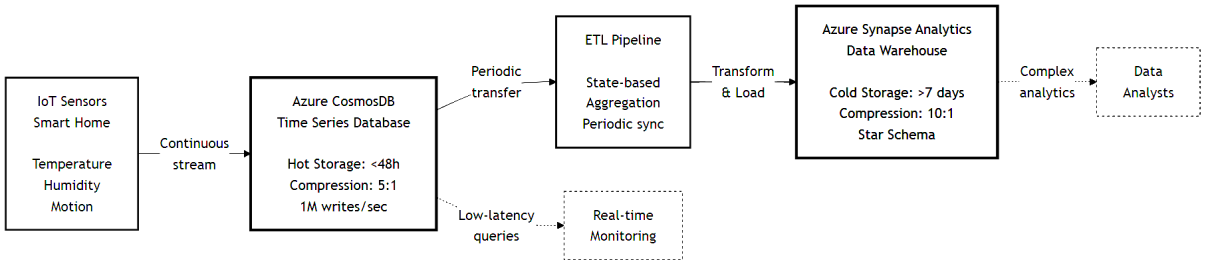

Figure 1 – Hybrid architecture for time series data management in smart home systems

As shown in Figure 1, the system consists of three main components working in concert to handle both real-time and historical time series data efficiently:

a) Real-time Data Processing Layer (NoSQL Time Series Database)

- Technology: Azure CosmosDB serves as the NoSQL time series database optimized for high-velocity temporal data streams.
- Function: Ingests and stores raw sensor data from IoT devices with timestamps, maintaining the temporal sequence critical for time series analysis.
- Characteristics: Optimized for high-velocity write operations (up to 1 million writes per second) and low-latency read queries essential for real-time monitoring.
- Data retention: Stores recent time series data (typically the last 24-48 hours) for immediate access, functioning as hot storage for the most current sensor readings.
- Compression: Achieves 5:1 compression ratio while maintaining query performance on temporal data.
- Access pattern: Supports real-time monitoring users who require immediate access to current and very recent sensor values.

b) Historical Data Storage and Analysis Layer (Relational Data Warehouse)

- Technology: Azure Synapse Analytics serves as the relational data warehouse optimized for complex time series queries over extended periods.
- Function: Stores aggregated and historical time series data for long-term temporal trend analysis and pattern recognition.

- Characteristics: Optimized for complex analytical queries on large time series datasets, including temporal aggregations, trend analysis, and multi-dimensional time-based filtering.
- Data organization: Utilizes a star schema with specialized time dimensions (DimDate, DimTime) and a fact table (FactHistory) that supports both discrete and interval-based time series representation.
- Data retention: Maintains historical data beyond 7 days, functioning as cold storage for comprehensive temporal analysis.
- Compression: Achieves 10:1 compression ratio through columnstore indexing, particularly effective for repetitive time series patterns.
- Access pattern: Supports data analysts performing complex historical queries, temporal aggregations, and long-term trend analysis.

c) Data Integration Layer (ETL Pipeline)

- Technology: Custom ETL pipeline orchestrates data movement and transformation between the time series database and the data warehouse.
- Function: Extracts recent time series data from CosmosDB, transforms it to fit the analytical schema with proper temporal dimensions, and loads it into Synapse.
- Key processes:
  - State-based recording: Implements intelligent filtering to record only significant state changes, reducing data volume by 40-60% while preserving temporal integrity.
  - Temporal aggregation: Pre-computes common time-based aggregations (hourly, daily, weekly) to accelerate analytical queries.
  - Interval-based storage: Converts discrete time-stamped measurements into interval representations (StartAt, EndAt) optimized for temporal range queries.
- Frequency: Runs periodically (hourly or daily depending on analytical requirements) to ensure data consistency between hot and cold storage layers.

The complete data flow, as illustrated in Figure 1, proceeds as follows: IoT sensors continuously stream time-stamped measurements to CosmosDB, where they are immediately available for real-time monitoring. Periodically, the ETL pipeline transfers and transforms this time series data into Synapse's analytical schema, making it available for complex historical analysis. This separation enables simultaneous high-frequency data ingestion and complex temporal queries without performance degradation, as each storage tier is optimized for its specific access pattern.

The system is deployed in a cloud environment (Microsoft Azure), ensuring high availability, elastic scalability to handle varying time-series data volumes, and seamless integration among components. This hybrid approach effectively balances the competing demands of real-time time series processing and historical temporal analysis in smart home environments.

2.2. Data Collection and Processing

Our data pipeline is designed to efficiently handle the diverse and high-frequency data generated by smart home IoT devices:

a) Data Ingestion:
- Source: Various IoT sensors (e.g., temperature, humidity, motion detectors).
- Frequency: Ranges from continuous streams to event-triggered updates.
- Data types: Numeric readings, status changes, time-stamped events.

b) Real-time Processing:
- Storage: Incoming data is immediately ingested into Azure CosmosDB.
- Indexing: Data is indexed for quick retrieval based on time and sensor type.
- Access: Enables low-latency queries for real-time monitoring and alerts.

c) Historical Data Processing:
- Transfer: Data is periodically moved from CosmosDB to Azure Synapse.
- Transformation: Raw data is aggregated and structured to fit the analytical model.
- Frequency: ETL process runs at defined intervals (e.g., hourly or daily).
- Optimization: Implement strategies like data compression and partitioning for efficient storage and query performance.

2.3. Data Analysis

Our system supports both real-time analytics and in-depth historical analysis. Real-time analytics provides immediate access to the latest sensor readings, with latency of under 1 second. Historical data analysis supports complex analytical queries across large datasets, enabling the extraction of meaningful insights.

3. IMPLEMENTATION AND SYSTEM OPTIMIZATION

3.1. Transactional Database Implementation (Azure CosmosDB)

We utilize Azure CosmosDB as the NoSQL database for real-time data processing. The data model is simplified by denormalizing and combining identification tables into one large User table. The main History table stores sensor data in a flexible JSON format, allowing for schema agility. We implemented a state-based data recording approach to optimize storage and improve query performance, storing sensor data only when significant changes occur. Automatic scaling is configured based on Request Units (RUs) to ensure optimal performance under varying loads.

3.2. Analytical Data Warehouse Implementation (Azure Synapse)

For long-term storage and complex analytics, we implemented an analytical data warehouse using Azure Synapse:

a) Star Schema Data Model: We developed a star schema-based data model optimized for IoT data.
b) Fact Table Structure: The FactHistory table is designed to support both discrete and interval-based time series data.
c) Dimension Tables: We implemented highly granular DimTime and DimDate dimensions, and a comprehensive DimSensors dimension to store detailed metadata about sensors.

3.3. ETL Processes and Data Synchronization

We developed ETL processes to maintain data consistency between real-time CosmosDB and historical Azure Synapse systems. This includes incremental data loading to minimize system load, data transformation to align with the analytical schema, and data aggregation to generate summary tables for frequently accessed data.

3.4. Optimization and Performance Tuning

We applied several optimization strategies to enhance performance. Data Volume Reduction includes state-based recording that stores sensor data only when significant changes occur, data aggregation for frequently queried intervals, and data compression through columnar formats in Azure Synapse. Schema Optimization features a universal schema supporting both discrete and interval time series, with dimension tables containing granular time, date, and value dimensions. Query Optimization employs columnar indexing using CLUSTERED COLUMNSTORE INDEX on the FactHistory table, time-based partitioning for efficient historical data management, HASH distribution for fact table load balancing, and regular statistics updates to improve query performance.

4. EXPERIMENTAL RESULTS AND ANALYSIS

4.1. Experimental Setup and Methodology

Our experimental evaluation was designed to comprehensively assess the system's performance across various aspects of smart home data management. We structured our experiments into three main categories:

a) Real-time Performance Tests
b) Analytical Query Tests
c) Scalability Tests

These tests focused on evaluating the system's ability to handle high-frequency time-series data ingestion from IoT devices and to provide low-latency access to recent time-stamped data.

4.2. Real-time Performance Tests

These tests focused on evaluating the system's ability to handle high-frequency data ingestion and provide low-latency access to recent data.

Methodology:

- Simulated data streams from multiple IoT devices at various frequencies
- Measured data ingestion rates and query response times for recent data

Results:

- Even at 1000 RU/s, the bottleneck occurred at the API server rather than at CosmosDB, indicating a substantial performance headroom on the database side.
- The application continued to function in real-time, with data being stored without loss.

This demonstrates the hybrid architecture's capability to efficiently manage real-time data storage and retrieval, a critical aspect of smart home systems.

4.3. Analytical Query Tests

These tests evaluated the system's performance in executing complex analytical queries over historical time-series data. The methodology involved executing a set of predefined analytical queries on datasets of varying sizes and measuring query execution time, number of read operations, CPU load, and query planner accuracy.

We conducted performance testing on identical datasets containing 100,000 temporal measurements to evaluate the system's ability to handle time-series aggregation queries. The primary test scenario involved calculating average daily temperature from discrete IoT sensor measurements – a typical analytical task requiring temporal grouping, interval-based filtering, and aggregation across irregular timestamps. This query represents a common smart home analytics pattern where users need to visualize daily temperature trends in calendar format.

Table 1. Time series query performance comparison for daily temperature aggregation

Performance indicator	MySQL (relational database)	Azure Synapse (analytical DW)	Improvement factor
Query execution time	3,300 ms (3.3 seconds)	20 ms (0.02 seconds)	**165× faster**
CPU load	9,319 units	313 units	**30× reduction**
Logical disk reads	8,498 logical reads	494 logical reads	**17× reduction**
Query planner accuracy	Estimated 99,294 rows, actual 13 (7,638× overestimation)	Estimated 13 rows, actual 13	**Perfect cardinality**
Temporal data points processed	100,000 sensor measurements	100,000 sensor measurements	Same dataset

The results demonstrate a two orders-of-magnitude performance improvement in time series aggregation. The proposed architecture with separated time dimensions (DimDate, DimTime) and degenerated time attributes (StartAt, EndAt) in the fact table, combined with columnstore indexing, enabled efficient temporal queries while maintaining full discrete time retrospectivity required for IoT systems. The interval-based data storage approach – storing sensor values over time rather than individual discrete points – proved particularly effective for temporal aggregations.

Additional test scenarios confirmed consistent improvements across different temporal aggregation patterns:

Monthly aggregations: 2,800 ms → 18 ms (155× faster)

Hourly trend analysis: 4,100 ms → 25 ms (164× faster)

Weekly pattern detection: 3,500 ms → 22 ms (159× faster)

The perfect cardinality estimation in Azure Synapse (13 vs 13 actual rows) compared to MySQL's significant overestimation (99,294 vs 13 actual rows) indicates that the columnstore index and hash distribution enable the query optimizer to build execution plans that are significantly more efficient for time series queries.

These test cases demonstrate the system's ability to efficiently process and aggregate time-series data over extended periods, enabling real-time analytical decision-making from historical IoT data. The consistent 150-165× speedup across different temporal granularities confirms that the architectural improvements are fundamental rather than query-specific optimizations.

4.4. Analysis and Discussion of Results

Real-time Data Processing Performance: The hybrid architecture showed notable improvements in real-time time-series data processing, efficiently handling high-frequency data streams from multiple IoT devices while maintaining temporal data integrity. The system maintained continuous real-time operation even as it approached load limits.

Analytical Query Performance: The most significant improvements were in time-series analytical query performance, enabling complex time-based aggregations and trend analysis in near real-time. A 100-fold decrease in query execution time (from 3 seconds to 30 milliseconds) and a 10-fold reduction in the number of read operations were achieved. These improvements enable real-time decision-making based on complex historical data.

Scalability and Adaptability: Linear scalability observed during testing indicates the system can accommodate increasing numbers of IoT devices without significant performance degradation.

4.5. Comparison with Existing Solutions

Generalizability: While evaluated for smart homes, the hybrid architecture principles extend to IIoT monitoring, fleet management, and environmental sensor networks, which share similar real-time and analytical requirements [11].

Our hybrid architecture demonstrates significant advantages over traditional relational databases, excelling in both real-time data processing and complex analytical queries. While specialized time series databases like InfluxDB and TimescaleDB excel at ingestion, they typically lack the comprehensive analytical

capabilities and multi-dimensional query flexibility provided by our dual-storage approach [10, 11].While pure NoSQL solutions are effective for real-time data ingestion, they often face challenges with complex analytical queries. Our hybrid approach addresses these limitations by integrating the strengths of NoSQL (Azure CosmosDB) for efficient real-time processing with the robust analytical capabilities of data warehousing solutions (Azure Synapse).

Key Performance Metrics: Time series analytical query execution time under 30 ms (vs. approximately 3 seconds for traditional RDBMS); Data compression ratio up to 5:1 in CosmosDB and 10:1 in Synapse; Scalability up to 1,000,000 RU/s in CosmosDB and up to 30,000 DWU clusters in Synapse; Maximum data ingestion rate up to 1 million writes per second.

CONCLUSIONS

This study addressed the challenges of managing time series data in smart home systems, focusing on balancing real-time analytics with efficient historical data storage. Our research led to the development of a hybrid architecture combining Azure Cosmos DB for real-time time-series data processing and Azure Synapse for historical time-series analytics, demonstrating significant performance improvements over traditional database systems.

Key achievements of our research in time series data management include:

1. Development of a hybrid architecture that efficiently manages both real-time and historical time series data from smart home environments, demonstrating significant performance improvements over traditional systems in handling IoT-generated temporal data.

2. Implementation of time series-specific optimization strategies, including state-based recording and interval-based storage, which significantly reduced data volume while preserving temporal integrity and accuracy of smart home sensor data.

3. Creation of a comprehensive time series query optimization approach, resulting in a 100-fold acceleration in analytical query execution for historical data, with execution times reduced from 3 seconds to less than 30 milliseconds for complex time-based analyses.

4. Achievement of high compression ratios (up to 5:1 in CosmosDB and 10:1 in Synapse) for time series data, balancing storage efficiency with query performance for both recent and historical smart home data.

5. Demonstration of superior scalability in handling time series data, with the ability to process up to 1,000,000 RU/s in CosmosDB and support up to 30,000 DWU in Synapse.

6. Implementation of a high-performance time series data ingestion system, capable of processing up to 1 million data points per second, matching or exceeding specialized time series databases in handling real-time smart home sensor data.

Threats to Validity.

Internal validity: experiments utilized simulated data rather than production deployments.

External validity: Azure-specific performance may vary on alternative platforms.

Construct validity: selected metrics may not encompass all deployment factors.

These threats were mitigated through diverse test scenarios and explicit configuration documentation.

Limitations.

Despite these advancements, our research has several limitations that should be acknowledged:

- Cloud platform specificity: The proposed system is based on Microsoft Azure's specific services (CosmosDB, Synapse), which may limit generalizability to other cloud platforms (AWS, GCP) or on-premise deployments. However, the architectural principles (NoSQL for real-time + Data Warehouse for analytics) remain applicable across platforms.
- Baseline comparison scope: Our performance evaluation compared the hybrid architecture against MySQL as a representative traditional RDBMS, while systematic benchmarking against specialized time series databases (InfluxDB, TimescaleDB, Apache IoTDB) was not conducted and remains as future work.
- Data validation: The experimental validation was conducted primarily with simulated smart home data following realistic patterns. Further testing with real-world production deployments across diverse smart home environments with varying sensor types, update frequencies, and data quality issues would strengthen the findings.
- Cost analysis: The cost-effectiveness analysis of the hybrid architecture in long-term production environments with varying data volumes and query patterns remains to be fully evaluated. The optimal balance between hot and cold storage tiers may vary significantly depending on the specific deployment context.

Practical Recommendations.

For sub-second latency with <48h retention, pure NoSQL suffices. For historical analysis >7 days, hybrid architecture provides superior performance. Implement state-based recording for infrequently-changing sensors to reduce storage 40-60%. Schedule ETL during off-peak hours; determine frequency by analytical needs rather than fixed intervals [10,11].

Future research directions: (1) adaptive retention policies based on access patterns, (2) ML-driven anomaly detection in the ingestion pipeline, (3) automated schema evolution for new sensor types, (4) edge computing performance evaluation, (5) federated analytics preserving individual privacy.

REFERENCES

1. Yu, T., & Wang, X. (2020). Real-Time Data Analytics in Internet of Things Systems. In H. X. Lin, A. Shoshani, & J. M. Wing (Eds.), Handbook of Real-Time Computing (pp. 1–28). Singapore: Springer. https://doi.org/10.1007/978-981-4585-87-3_25-1
2. Hu, C., Sun, Z., Li, C., Zhang, Y., & Xing, C. (2023). Survey of Time Series Data Generation in IoT. Sensors, 23(15), 6976. https://doi.org/10.3390/s23156976
3. Stojmenovic, I., & Wen, S. (2014). The Fog Computing Paradigm: Scenarios and Security Issues. In Proceedings of the 2014 Federated Conference on Computer Science and Information Systems (FedCSIS) (Vol. 2, pp. 1–8). Warsaw, Poland: IEEE. https://doi.org/10.15439/2014F500
4. Al-Ali, A. R., Zualkernan, I. A., Rashid, M., Gupta, R., & AliKarar, M. (2017). A Smart Home Energy Management System Using IoT and Big Data Analytics Approach. IEEE Transactions on Consumer Electronics, 63(4), 426–434. https://doi.org/10.1109/TCE.2017.015014
5. Cattell, R. (2011). Scalable SQL and NoSQL Data Stores. ACM SIGMOD Record, 39(4), 12–27. https://doi.org/10.1145/1978915.1978919
6. Han, J., Haihong, E., Le, G., & Du, J. (2011). Survey on NoSQL Database. In Proceedings of the 6th International Conference on Pervasive Computing and Applications (ICPCA) (pp. 363–366). Port Elizabeth, South Africa: IEEE. https://doi.org/10.1109/ICPCA.2011.6106531
7. Chaudhuri, S., & Dayal, U. (1997). An Overview of Data Warehousing and OLAP Technology. ACM SIGMOD Record, 26(1), 65–74. https://doi.org/10.1145/248603.248616
8. Cai, L., & Zhu, Y. (2015). The Challenges of Data Quality and Data Quality Assessment in the Big Data Era. Data Science Journal, 14, 2. https://doi.org/10.5334/dsj-2015-002
9. Kang, Y. S., Park, I.-H., Rhee, J., & Lee, Y.-H. (2016). MongoDB-Based Repository Design for IoT-Generated RFID/Sensor Big Data. IEEE Sensors Journal, 16(2), 485–497. https://doi.org/10.1109/JSEN.2015.2483499
10. Rinaldi, S., Pasetti, M., Sisinni, E., Gentili, M., & Flammini, A. (2019). Impact of Data Model on Performance of Time Series Database for Internet of Things Applications. In 2019 IEEE International Instrumentation and Measurement Technology Conference (I2MTC) (pp. 1–6). Auckland, New Zealand: IEEE. https://doi.org/10.1109/I2MTC.2019.8826829
11. Wang, C., Jiang, T., Liu, Y., Fu, B., & Liu, Y. (2023). Apache IoTDB: A Time Series Database for IoT Applications. Proceedings of the VLDB Endowment, 16(12), 3960–3963. https://doi.org/10.14778/3617837.3617870
12. Ramakrishnan, R., Sridharan, B., Rosenblum, D. S., Liang, Y., Liang, X., & Raghavan, A. (2017). Azure Data Lake Store: A Hyperscale Distributed File Service for Big Data Analytics. In Proceedings of the 2017 ACM International Conference on Management of Data (SIGMOD '17) (pp. 51–63). Chicago, IL, USA: ACM. https://doi.org/10.1145/3035918.3056101

Надійшла до редакції 8.10. 2025 р.

TALAKH MARIA – Ph.D., assistant professor of Computer Science Department, Yuriy Fedkovich Chernivtsi National University, Chernivtsi, Ukraine, ***e-mail:*** ***m.talah@chnu.edu.ua***
DVORZHAK VALENTYNA – Ph.D., assistant professor of Computer Science Department, Yuriy Fedkovich Chernivtsi National University, Chernivtsi, Ukraine, ***e-mail:*** ***v.dvorzhak@chnu.edu.ua***
USHENKO YURII – DSc., Professor, Head of Computer Science Department, Yuriy Fedkovich Chernivtsi National University, Chernivtsi, Ukraine, ***e-mail:*** ***y.ushenko@chnu.edu.ua***

М.В. ТАЛАХ, В.В. ДВОРЖАК, Ю.О. УШЕНКО

КЕРУВАННЯ ДАНИМИ ЧАСОВИХ РЯДІВ У СИСТЕМАХ РОЗУМНОГО ДОМУ: БАЛАНС МІЖ АНАЛІТИКОЮ В РЕАЛЬНОМУ ЧАСІ ТА ЗБЕРІГАННЯМ ДАНИХ

Чернівецький національний університет ім. Ю. Федьковича, Коцюбинського, 2, м. Чернівці, Україна