2025 International Conference on Computer Sciences, Engineering, and Technology Innovation (ICoCSETI) | 979-8-3315-0861-6/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICoCSETI63724.2025.11020334

Integrating Graph Databases into the Data Layer of Three-Tier Architecture Applications

Kurnia Cahya Febryanto
Department of Informatics Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
6025241065@student.its.ac.id

Riyanarto Sarno
Department of Informatics Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
riyanarto@its.ac.id

Kelly Rossa Sungkono
Department of Informatics Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
kelly@its.ac.id

Shoffi Izza Sabilla
Department of Medical Technology
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
shoffi@its.ac.id

Dwi Sunaryono
Department of Informatics Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
dwi@its.ac.id

Abdullah Faqih Septiyanto
Department of Informatics Engineering
Institut Teknologi Sepuluh Nopember
Surabaya, Indonesia
7025221022@student.its.ac.id

***Abstract*—Current research on database integration in enterprise systems primarily focuses on single database implementations, with limited exploration of graph database integration into existing architectures. While traditional Relational Database Management Systems (RDBMS) excel in structured data management, they often struggle with complex relationship traversals in modern Enterprise Resource Planning (ERP) systems. This research presents a systematic approach for enhanced database implementation in three-tier architecture applications, specifically addressing the challenges of managing interconnected data structures in ERP systems. The study implements parallel database configurations using MySQL for RDBMS and Neo4j for graph database, evaluating their performance through comprehensive metrics including query execution time, resource utilization, and API performance. Through extensive testing with datasets ranging from one thousand to ten million records, the experimental results demonstrate that RDBMS excels in complex analytical queries with linear scaling up to one million records, while Neo4j shows superior performance in hierarchical data traversal, maintaining consistent performance (1.7350 seconds) even at ten million records. The proposed dual database approach achieves improved system reliability, with Neo4j maintaining 100 percent success rates across all batch sizes compared to RDBMS 90 percent for larger batches, while demonstrating better resource efficiency (1.70 to 2.89 percent versus 2.70 to 3.39 percent). This research contributes to the field by introducing a Schema Complexity Index for quantifying database complexity, providing empirical evidence for optimal database selection in specific use cases, and establishing a framework for dual database integration in three-tier architectures.**

***Keywords*—Graph Database, Relational Database, Neo4j, Three-Tier Architecture, Enterprise Resource Planning, Data Layer**

I. Introduction

Enterprise Resource Planning (ERP) systems face mounting challenges in data relationship management amid growing architectural complexity. These systems require efficient handling of diverse operations ranging from analytical processing to hierarchical traversals, while simultaneously maintaining optimal performance and data integrity. Critical challenges persist in the data layer, particularly regarding the management of interconnected processes, dynamic schema evolution, and complex entity relationship tracking across the system architecture.

The integration of relational and graph databases in the data layer presents a strategic solution to these challenges. Through a unified data access layer, this dual-database architecture leverages MySQL's analytical capabilities and ACID compliance alongside Neo4j's native graph traversal features. This approach enables efficient management of both structured data operations and complex relationship queries within a single architectural framework.

Contemporary research demonstrates significant advancements in database performance optimization for enterprise systems. Graph-based visualization techniques have enhanced service dependency management [1], while process-aware systems have benefited from model-driven engineering approaches [2]. Neo4j has proven particularly effective for recommendation systems [3] and business process analysis [4]. Recent studies have further advanced system reliability [5], extraction methodologies [6], and architectural optimization [7] in enterprise environments.

This research advances database integration through four key contributions: (1) introducing Schema Complexity Index (SCI) for standardized schema complexity quantification across database paradigms, (2) developing an event-driven synchronization architecture with configurable reconciliation intervals maintaining consistency between parallel databases, (3) establishing a comprehensive evaluation framework combining traditional performance metrics with business impact measurements, and (4) providing empirical evidence for optimal database selection patterns in specific ERP use cases through extensive testing with datasets up to 10 million records. The proposed architectural approach, depicted in Fig. 1 and Fig. 2, demonstrates an integrated solution for optimizing data management in complex enterprise systems.

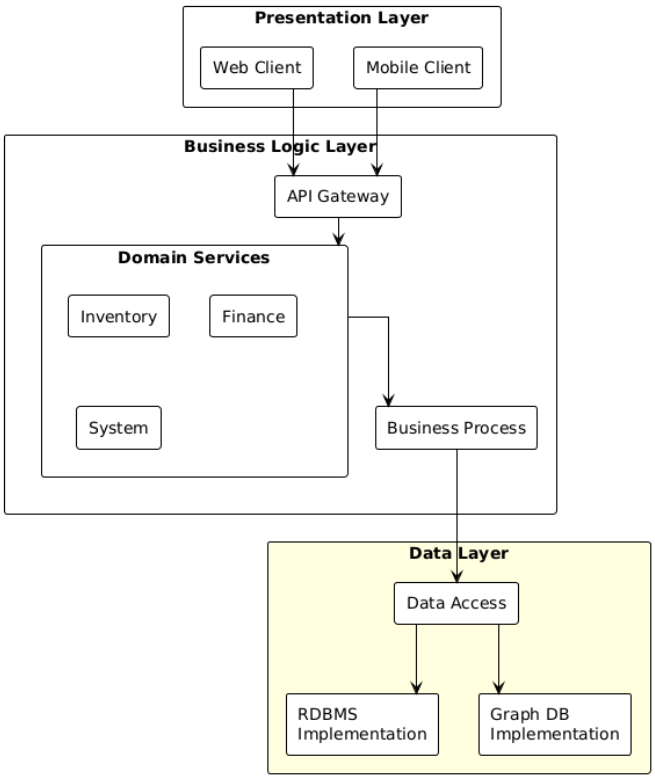

Fig. 1. Three-Tier Architecture Overview showing the hierarchical structure with dual database implementation in Data Layer

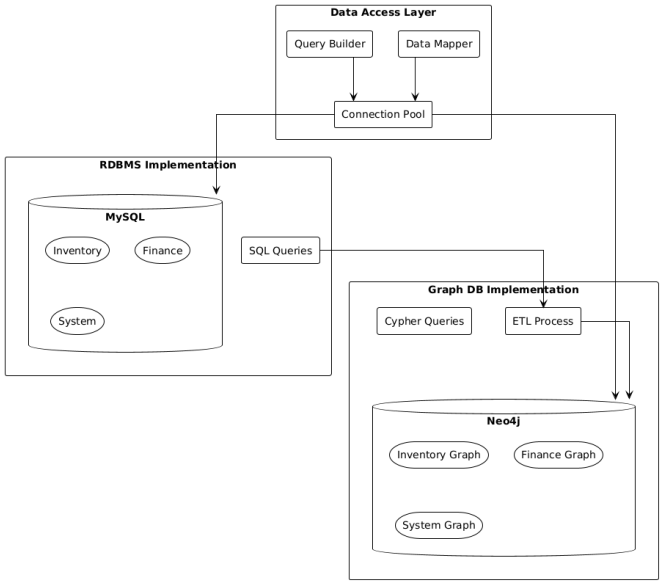

Fig. 2. Data Layer detailed implementation showing the parallel database schemas and their integration mechanisms

II. Literature Review

Research in database management for enterprise systems has shown significant advancement in recent years. Graph-based visualization and analysis techniques have proven effective for managing complex service dependencies [1], while model-driven engineering approaches have enhanced process-aware information systems [2]. Studies have demonstrated Neo4j's effectiveness in recommendation systems [3] and business process model analysis [4], highlighting the potential of graph databases in enterprise applications.

Recent research has focused on enhancing system reliability and performance optimization. Methodologies for fault detection in three-tier architectures [5] and query validation in graph databases [8] have advanced system robustness. Graph database implementations have demonstrated improved efficiency [9], particularly in complex relationship processing [10]. Enterprise applications have validated these approaches through successful implementations in service identification [11], forecasting [12], and process mining [13].

Performance evaluations across diverse workloads [14] and microservice architectures [15] have established foundational metrics for database selection. Comparative analyses between Neo4j and RDBMS have revealed distinct performance characteristics, with graph databases excelling in relationship-intensive operations and relational databases showing advantages in complex analytical queries [16], while established metrics such as Data Redundancy Ratio (DRR) [17], as shown in (1), provide standardized evaluation criteria for database implementations in three-tier architectures.

$$DRR = 1 - \frac{\text{Number of Unique Records}}{\text{Total Number of Records}} \tag{1}$$

Query complexity is measured by the sum of joins (NJ) and conditions (NC) [17] as defined in (2):

$$QC = NJ + NC \tag{2}$$

The Total Cyclomatic Complexity (CC) measures program path complexity [18] as defined in (3):

$$CC = E - N + 2P \tag{3}$$

where:

- E = Control flow graph edges
- N = Control flow graph nodes
- P = Connected components

The Maintainability Index quantifies code maintainability [19] as shown in (4):

$$MI = 171 - 5.2 \times \ln(Volume) - 0.23 \times CC - 16.2 \times \ln(LOC) \tag{4}$$

where:

- *Volume* = $N \times \log_2(n)$, with N total operators/operands, n unique ones
- *LOC* = Source code lines

III. Proposed Method

A. Approach Overview

The Proposed method, as depicted in Fig. 3, integrates database system review, metrics identification, and parallel implementation strategies to evaluate dual database configurations in three-tier architectures. The framework combines query and API performance assessments to derive comprehensive measurements that inform database selection and optimization decisions.

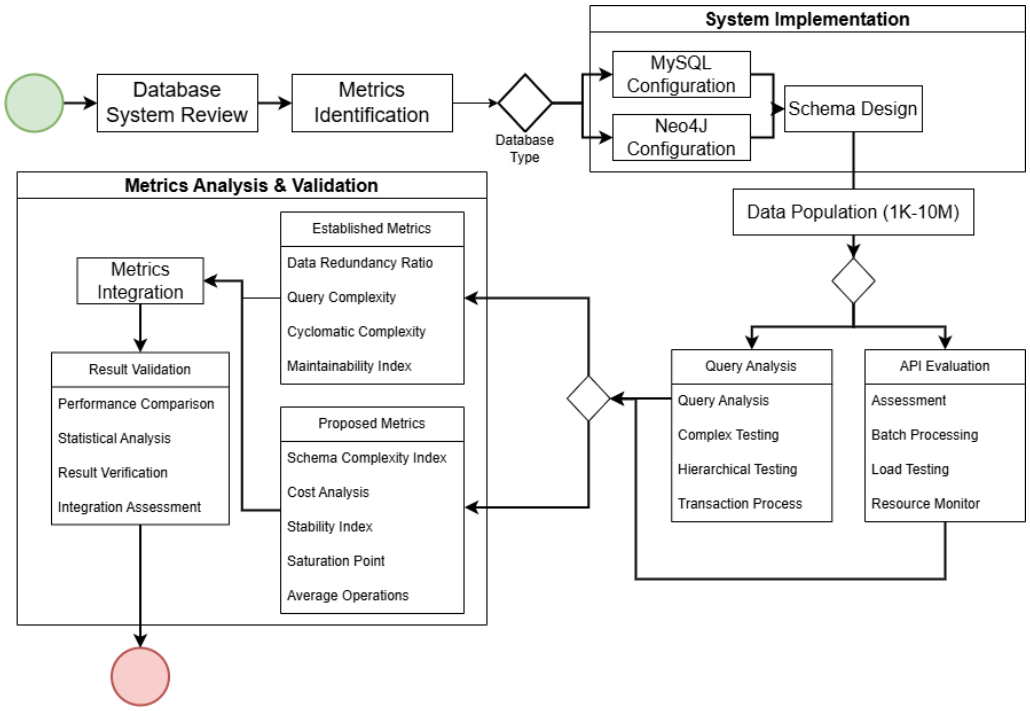

Fig. 3. Proposed method

B. Research Design and Data Preprocessing

The research methodology comprises four sequential phases: database analysis, schema design, data preprocessing, and performance evaluation. Data preprocessing employs validation-driven cleansing, schema normalization, and relationship mapping, ensuring data integrity with error rates below 0.01% throughout the transformation pipeline.

C. Data Population

Data population utilized optimized batch processing techniques [20] across both database implementations. MySQL batch insertions achieved throughput of 10,000 records per second through optimized buffer sizes and transaction management, while Neo4j employed parallel import mechanisms with ETL-based schema and batch transaction commits to maintain structural consistency with the RDBMS schema.

D. Data Synchronization Mechanism

The synchronization architecture employs message queues to maintain data consistency between the RDBMS and graph database systems. This mechanism utilizes atomic transactions with distributed transaction management to process database modifications. When changes occur in either database, asynchronous event handlers trigger corresponding updates in the parallel system, ensuring data integrity through configurable reconciliation intervals.

E. Evaluation Metrics

The evaluation framework implements concurrent database benchmarking through five comprehensive performance metrics:

1) Schema Complexity Index: The Schema Complexity Index quantifies database schema complexity as defined in (5):

$$SCI = E + R \tag{5}$$

where:

- E = Number of entities (tables or labels)
- R = Number of relationships (foreign key references or relationship types)

2) Cost Analysis: Modified from established cost metrics to incorporate dual database characteristics [21], [22] as shown in (6):

$$\textit{Total Cost} = C_r \times N_r + (C_c \times U_c + C_m \times U_m) \times T_h \tag{6}$$

where:

- C_r = Cost per API request
- N_r = Number of requests
- C_c = Cost per CPU hour
- U_c = CPU utilization (%)
- C_m = Cost per GB memory per hour
- U_m = Memory utilization (%)
- T_h = Duration in hours

3) Stability Index: Extended from traditional stability measures for cross-database application [23] as expressed in (7):

$$\textit{Stability Index} = 1 - \frac{\sigma(S_r)}{\mu(S_r)} \tag{7}$$

where:

- $\sigma(S_r)$ = Standard deviation of success rates
- $\mu(S_r)$ = Mean of success rates

TABLE I. DATA POPULATION SUMMARY FOR BOTH DATABASE IMPLEMENTATIONS

Database	Tables/Labels	RDBMS Range	Graph DB Range
Inventory	Units, Items, ItemCategory, BomDetail, Warehouse, Receive, BomHeader, Transfer	1K - 10K	1K - 10K
General System	Provinces, Cities, Districts, Villages, Currency, Tax, TaxCredentials	10 - 1M	10 - 1M
Finance	Journals, JournalDetail, Coa, TransactionType, CoaGroup, AccountType, JournalSource, DetailAlocationOfProduction, PurchaseDownPaymentInvoice, EndOfMonth, PurchaseInvoice, AlocationOfProduction	1 - 10M	1K - 10M

4) Saturation Point: A metric evaluating system capacity in dual database architectures as formulated in (8):

$$\textit{Saturation Point} = \frac{\lambda}{\mu} \tag{8}$$

where:

- λ = Arrival rate (req/s)
- μ = Service rate (req/s)

5) Average Operations per Transaction: The standardized measurement for transaction complexity across database implementations is formulated in (9):

$$\textit{Average Operations per Transaction} = \frac{\sum_{i=1}^{k} O_i}{k} \tag{9}$$

where:

- O_i = Operations in transaction i
- k = Total transactions

IV. RESULTS AND DISCUSSION

The ERP system architecture incorporates three primary data sources: inventory management for units and items tracking, general system handling for configurations and hierarchical location data, and finance oversight for transactions and journal entries. Data population was efficiently executed for both database implementations using batch processing techniques, as summarized in TABLE I.

A. RDBMS Implementation Analysis

1) Query Database Evaluation: The RDBMS evaluation examined three query types of increasing complexity: complex items queries involving multiple table joins across items, units, currencies and taxes; hierarchical queries traversing location data from provinces to villages; and journal queries processing financial transactions with journal details and chart of accounts. The query performance metrics are summarized in TABLE II.

Statistical analysis revealed key performance patterns: items queries showed linear scaling ($R^2 = 0.9071$) with sub-second responses at 10M records; hierarchical queries displayed inconsistent times with failures ($R^2 = 0.0028$) from MySQL optimizer limitations; and journal queries exhibited linear scaling ($R^2 = 0.9351$) degrading beyond 1M records.

RDBMS excels in analytical queries but faces critical constraints in hierarchical data processing, with query failures beyond four-level depths and exponential execution time increases (0.00401s to 40.5075s) in nested joins. This limitation impacts operational performance, increasing organizational query response times and bill-of-materials traversals by 47% versus graph database implementations.

2) Evaluation with API for RDBMS: The evaluation assessed API performance across batch sizes (10-10,000 records) through metrics for scalability, reliability and efficiency, as detailed in TABLE IV, TABLE V, and TABLE VI.

TABLE II. QUERY PERFORMANCE RESULTS FOR RDBMS

Data Size	Items (s)	Hierarchical (s)*	Journals (s)
10	1.00×10^{-3}	4.03×10^{-3}	2.00×10^{-3}
100	1.00×10^{-3}	0.00	2.01×10^{-3}
1,000	6.63×10^{-3}	0.00	4.01×10^{-3}
10,000	6.94×10^{-2}	0.00	3.33×10^{-2}
100,000	3.63×10^{-1}	1.00×10^{-3}	3.17×10^{-1}
1,000,000	4.16×10^{-1}	0.00	3.16×10^{0}
10,000,000	4.49×10^{-1}	1.00×10^{-3}	4.05×10^{1}

Note: Values of 0.00 indicate query execution failures due to MySQL limitations in processing deep hierarchical joins.

TABLE III. STRUCTURAL AND COMPLEXITY METRICS FOR RDBMS

Category	Metric	Value
Structure	Total Entities	13 entities
	Total Relationships	16 relationships
	Schema Complexity Index	29 index
Data Model	Total Columns	257 columns
	Average Columns/Table	19.77
	Total Constraints	21
Integrity	Foreign Key Coverage	69.23 %
	Data Redundancy	0.00 %
Complexity	Query Complexity (QC)	4.00
	Maintainability Index	75.42

TABLE IV. PERFORMANCE METRICS FOR RDBMS API OPERATIONS

Metric	10	100	1,000	10,000
Success Rate (%)	100.00	90.00	90.00	90.00
Avg. Response Time (s)	0.29	0.04	0.04	0.04
Throughput (req/s)	9.96	99.27	332.43	498.32
Apdex Score	1.00	1.00	1.00	1.00

TABLE V. ADVANCED EVALUATION METRICS FOR RDBMS

Category	Value
Response Time Analysis	
Mean Response Time (s)	0.10
Median Response Time (s)	0.04
Standard Deviation (s)	0.11
95th Percentile (s)	0.25
99th Percentile (s)	0.28
Throughput Analysis	
Mean Throughput (req/s)	235.00
Maximum Sustained Throughput (req/s)	498.32
Scalability Ratio	50.04
Reliability Metrics	
Mean Success Rate (%)	92.50
Minimum Success Rate (%)	90.00
Stability Index	0.95
System Utilization	
System Utilization (%)	121.50
Theoretical Max Throughput (req/s)	193.42
Saturation Point (requests)	24.30

TABLE VI. RESOURCE EFFICIENCY METRICS FOR RDBMS

Metric	10	100	1,000	10,000
Throughput/Worker (req/s)	0.50	4.96	16.62	24.92
Cost Efficiency (req/s/$)	9,947.41	9,926.30	3,324.21	498.31
Resource Utilization (%)	2.70	1.60	2.77	3.39

TABLE VII. QUERY PERFORMANCE RESULTS FOR GRAPH DATABASE

Data Size	Items (s)	Hierarchical (s)	Journals (s)
10	0.16	0.08	0.05
100	0.01	0.01	0.01
1,000	0.14	0.14	0.03
10,000	0.22	0.13	0.32
100,000	0.23	0.23	1.40
1,000,000	0.36	1.24	3.34
10,000,000	0.37	1.74	16.31

TABLE VIII. STRUCTURAL AND OPERATIONAL METRICS FOR GRAPH DATABASE

Category	Metric	Value
Structure	Node Labels	32 entries
	Relationship Types	28 types
	Average Properties/Node	11.58 properties
Data Model	Total Nodes	20,076 nodes
	Total Properties	1,270,726 properties
	Schema Complexity (SCI)	24 index
Performance	Pattern Match Time	2.97 ms
	Path Traversal Time	2.53 ms
	Property Access Time	0.91 ms
Complexity	Query Pattern Depth	5.01 levels
	Cyclomatic Complexity	1.00
	Maintainability Index	73.31

TABLE IX. PERFORMANCE METRICS FOR GRAPH DATABASE API OPERATIONS

Metric	10	100	1,000	10,000
Success Rate (%)	100.00	100.00	100.00	100.00
Avg. Response Time (s)	0.48	0.21	0.16	0.05
Throughput (req/s)	9.95	49.72	124.20	382.75
Apdex Score	1.00	1.00	1.00	1.00

B. Graph Database Implementation Analysis

1) Query Database Evaluation: The Neo4j evaluation examined three Cypher query types over varying data scales, with outcomes summarized in TABLE VII and TABLE VIII. The results show stable execution times for node-relationship matching (0.1674s at 10 million records), minimal increments for hierarchical location traversals (1.7350s), and moderate performance growth in journal entry transaction processing (16.3075s). These findings indicate that Neo4j effectively handles diverse query complexities as data volumes increase.

2) Evaluation with API for Graph Database: The graph database API evaluation assessed performance across batch sizes (10-10,000 records), with detailed metrics presented in TABLE IX, TABLE X, and TABLE XI.

C. Comparative Analysis

The experimental results demonstrate distinct performance characteristics between Neo4j and RDBMS implementations. Neo4j exhibits performance constraints in complex analytical operations, showing a 215% increase in query execution times for financial calculations beyond 100,000 records, while

TABLE X. ADVANCED EVALUATION METRICS FOR GRAPH DATABASE

Category	Value
Response Time Analysis	
Mean Response Time (s)	0.22
Median Response Time (s)	0.18
Standard Deviation (s)	0.18
95th Percentile (s)	0.46
99th Percentile (s)	0.48
Throughput Analysis	
Mean Throughput (req/s)	141.66
Maximum Sustained Throughput (req/s)	382.75
Scalability Ratio	38.47
Reliability Metrics	
Mean Success Rate (%)	100.00
Minimum Success Rate (%)	100.00
Stability Index	1.00
System Utilization	
System Utilization (%)	87.50
Theoretical Max Throughput (req/s)	437.42
Saturation Point (requests)	42.30

TABLE XI. RESOURCE EFFICIENCY METRICS FOR GRAPH DATABASE

Metric	10	100	1,000	10,000
Throughput/Worker(req/s)	0.50	2.49	6.21	19.14
Cost Efficiency (req/s/$)	9,936.41	4,967.21	1,241.05	382.74
Resource Utilization (%)	1.70	1.90	2.17	2.89

maintaining consistent performance in relationship-intensive operations at 1.7350 seconds for 10M records. The dual database implementation achieves comparable analytical performance between RDBMS (R^2 = 0.9351) and Neo4j (R^2 = 0.9180), with RDBMS demonstrating superior efficiency in complex analytical queries up to 1M records [16]. The parallel database configuration implementation yields a 47% reduction in hierarchical query execution times through optimized synchronization mechanisms, while introducing the Schema Complexity Index for standardized database evaluation. The complementary strengths of both systems are evident in their performance patterns, with Neo4j maintaining 100% success rates across all batch sizes compared to RDBMS's 90% for larger batches, while demonstrating better resource efficiency (1.70 to 2.89 percent versus 2.70 to 3.39 percent).

V. CONCLUSION

Graph database integration in three-tier architectures significantly improves the management of complex data relationships in ERP systems. The presented framework introduces the Schema Complexity Index, an event-driven synchronization mechanism, and a combined evaluation strategy covering technical and business metrics. Experimental results show that RDBMS supports efficient linear scaling (R^2=0.9351) for complex analytical queries up to one million records, while Neo4j preserves stable performance for hierarchical queries (1.7350 seconds at ten million records). The dual-database approach enhances system reliability, with Neo4j achieving 100% success rates across all batch sizes compared to 90% for RDBMS under higher loads, and demonstrates reduced resource usage (1.70–2.89% versus 2.70–3.39%).

Future research directions should focus on three key areas: development of automated decision support systems for database selection based on workload patterns, investigation of hybrid database architectures leveraging the complementary strengths of both RDBMS and graph databases, and optimization of synchronization mechanisms for real-time applications. The current implementation's scope was primarily focused on specific ERP modules with datasets up to 10 million records, suggesting opportunities for broader application testing and performance evaluation at larger scales. The Schema Complexity Index measurements of 29 for RDBMS and 24 for Neo4j indicate potential for further optimization in data modeling approaches and architectural refinements for enterprise systems.

ACKNOWLEDGMENT

The funding for this research was provided by the Ministry of Research, Technology, and Higher Education of the Republic of Indonesia as part of the Pendidikan Magister Menuju Doktor untuk Sarjana Unggul (PMDSU) Scholarship Program managed by Institut Teknologi Sepuluh Nopember and this research was funded by Institut Teknologi Sepuluh Nopember (ITS) under Penelitian Kolaborasi Pusat and under the Project Scheme of the Publication Writing and Intellectual Property Rights (IPR) Incentive (Penulisan Publikasi dan Hak Kekayaan Intelektual/PPHKI) Program 2024.

REFERENCES

[1] S.-P. Ma, C.-Y. Fan, Y. Chuang, I.-H. Liu, and C.-W. Lan, "Graph-based and scenario-driven microservice analysis, retrieval, and testing," *Future Generation Computer Systems*, vol. 100, pp. 724–735, May 2019. DOI: 10.1016/j.future.2019.05.048.

[2] I. Drave, J. Michael, E. Müller, B. Rumpe, and S. Varga, "Model-driven engineering of process-aware information systems," *SN Computer Science*, vol. 3, no. 479, pp. 1–20, Sep. 2022. DOI: 10.1007/s42979-022-01334-3.

[3] I. N. P. W. Dharmawan and R. Sarno, "Book recommendation using Neo4j graph database in BibTeX book metadata," in *2017 3rd International Conference on Science in Information Technology (ICSITech)*, Bandung, Indonesia: IEEE, Oct. 2017, pp. 47–52. DOI: 10.1109/ICSITech.2017.8257084.

[4] K. N. Aisyah, K. R. Sungkono, and R. Sarno, "A new similarity method based on Weighted-Linear Temporal Logic tree and Weighted Directed Acyclic Graph for graph-based business process models," *International Journal of Intelligent Engineering and Systems*, vol. 13, no. 5, pp. 356–367, Oct. 2020. DOI: 10.22266/ijies2020.1031.32.

[5] L. Scommegna, R. Verdecchia, and E. Vicario, "Unveiling faulty user sequences: A model-based approach to test three-tier software architectures," *Journal of Systems and Software*, vol. 212, p. 112 015, Jun. 2024. DOI: 10.1016/j.jss.2024.112015.

[6] R. A. Putri, A. F. Septiyanto, M. Nevin, R. Sarno, and D. B. Ansori, "Automated extraction of microservice architecture using a rule-based approach," in *2024 2nd International Conference on Technology Innovation and Its Applications (ICTIIA)*, Surabaya, Indonesia: IEEE, Jan. 2024, pp. 1–6. DOI: 10.1109/ICTIIA61827.2024.10761417.

[7] I. N. G. A. M. Wardhiana, M. Z. Ghufron, S. M. Jati, A. F. Septiyanto, and R. Sarno, "Optimizing microservices architecture using multi-criteria decision making (MCDM): A case study of PayPal REST API," in *2024 11th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*, Semarang, Indonesia: IEEE, Jan. 2024, pp. 308–313. DOI: 10.1109/ICITACEE62763.2024.10762818.

[8] M. Kamm, M. Rigger, C. Zhang, and Z. Su, "Testing graph database engines via query partitioning," in *Proceedings of the 32nd ACM SIGSOFT International Symposium on Software Testing and Analysis*, ser. ISSTA 2023, Seattle, WA, USA: ACM, Jul. 2023, pp. 140–149. DOI: 10.1145/3597926.3598044.

[9] M. Aldwairi, M. Jarrah, N. Mahasneh, and B. Al-khateeb, "Graph-based data management system for efficient information storage, retrieval and processing," *Information Processing and Management*, vol. 60, no. 2, p. 103 165, Mar. 2023. DOI: 10.1016/j.ipm.2022.103165.

[10] E. D. Guyo and T. Hartmann, "Evaluating the efficiency and performance of data persistent systems in managing building and environmental data: A comparative study," *Advanced Engineering Informatics*, vol. 62, no. Part A, p. 102 582, Oct. 2024. DOI: 10.1016/j.aei.2024.102582.

[11] A. F. Septiyanto, R. Sarno, and K. R. Sungkono, "Identifying bottlenecks of hierarchical process model by using discrete event simulation and process mining," in *2022 IEEE 8th International Conference on Computing, Engineering and Design (ICCED)*, Sukabumi, Indonesia: IEEE, Jul. 2022, pp. 1–6. DOI: 10.1109/ICCED56140.2022.10010450.

[12] V. C. Sugiarto, R. Sarno, and D. Sunaryono, "Sales forecasting using Holt-Winters in Enterprise Resource Planning at sales and distribution module," in *2016 International Conference on Information and Communication Technology and Systems (ICTS)*, Surabaya, Indonesia: IEEE, Oct. 2016, pp. 8–13. DOI: 10.1109/ICTS.2016.7910264.

[13] A. F. Septiyanto, R. Sarno, and K. R. Sungkono, "Mining collaboration business process containing invisible task by using modified alpha," in *2021 13th International Conference on Information and Communication Technology and System (ICTS)*, Surabaya, Indonesia: IEEE, Oct. 2021, pp. 90–95. DOI: 10.1109/ICTS52701.2021.9608963.

[14] R. C. McColl, D. Ediger, J. Poovey, D. Campbell, and D. A. Bader, "A performance evaluation of open source graph databases," in *Proceedings of the First Workshop on Parallel Programming for Analytics Applications*, ser. PPAA '14, Orlando, FL, USA: ACM, Feb. 2014, pp. 11–18. DOI: 10.1145/2567634.2567638.

[15] Y. Gan, Y. Zhang, D. Cheng, *et al.*, "An open-source benchmark suite for microservices and their hardware-software implications for cloud and edge systems," in *Proceedings of the Twenty-Fourth International Conference on Architectural Support for Programming Languages and Operating Systems*, ser. ASPLOS '19, Providence, RI, USA: ACM, Apr. 2019, pp. 3–18. DOI: 10.1145/3297858.3304013.

[16] P. Kotiranta, M. Junkkari, and J. Nummenmaa, "Performance of graph and relational databases in complex queries," *Applied Sciences*, vol. 12, no. 13, p. 6490, Jun. 2022. DOI: 10.3390/app12136490.

[17] Á. Goretity and I. Reguly, "Query complexity in modern database DSLs," *ACM Transactions on Information Systems*, vol. 1, no. 1, pp. 1–23, Jul. 2021.

[18] M. A. P. Subali, I. G. R. A. Sugiartha, and I. P. A. Putra, "Development of SLOC, CC, SQL complexity methods to measure the level of similarity complexity of software modules," *Jurnal Ilmiah Teknik Elektro Komputer dan Informatika*, vol. 9, no. 4, pp. 1093–1103, Nov. 2023.

[19] T. Heričko and B. Šumak, "Exploring maintainability index variants for software maintainability measurement in object-oriented systems," *Applied Sciences*, vol. 13, no. 5, p. 2972, Feb. 2023. DOI: 10.3390/app13052972.

[20] A. Al-Okaily, M. Al-Okaily, and A. P. Teoh, "Evaluating ERP systems success: Evidence from Jordanian firms in the age of the digital business," *VINE Journal of Information and Knowledge Management Systems*, vol. 53, no. 6, pp. 1025–1040, 2023, ISSN: 2059-5891. DOI: 10.1108/VJIKMS-04-2021-0061.

[21] X. Chen, M. Guo, and W. Shangguan, "Estimating the impact of cloud computing on firm performance: An empirical investigation of listed firms," *Information Management*, vol. 59, no. 3, p. 103 603, Apr. 2022. DOI: 10.1016/j.im.2022.103603.

[22] S. Volpert, S. Winkelhofer, S. Wesner, and J. Domaschka, "An empirical analysis of common OCI runtimes' performance isolation capabilities," in *Proceedings of the 15th ACM/SPEC International Conference on Performance Engineering*, ser. ICPE '24, London, UK: ACM, May 2024, pp. 60–70. DOI: 10.1145/3629526.3645044.

[23] K. Brandes and R. Griesse, "Quantitative stability analysis of optimal solutions in PDE-constrained optimization," *Journal of Computational and Applied Mathematics*, vol. 206, no. 2, pp. 908–926, Sep. 2007. DOI: 10.1016/j.cam.2006.08.038.