

A Hybrid Strategy Sustainable for Combining Relational and Graph Databases to Improve the Effectiveness of Complex Relationship Analysis

HASSAN BEDIAR HASHIM
Middle Technical University,
Technical Trainers Institute,
Baghdad,
IRAQ

Abstract: - The purpose of this study is to develop and implement a hybrid approach that addresses the limitations of relational databases (such as MySQL) and graph databases (like Neo4j) in analyzing complex relationships. The study used an experimental methodology that replicated a setting with network data (recommendations, social ties) stored in a graph and structured data (customers, products) saved in a relational database. The findings showed that, while preserving data integrity, the suggested approach significantly reduced the execution time of complicated queries involving multi-level traversal of relationships (on average, 70% faster) when compared to depending only on the relational database. The study also showed that some data duplication resulted in a 15% increase in storage costs, which is acceptable given the notable improvements in speed. The study concludes that the hybrid model offers a workable option for applications that need dependable data traversal operations in addition to sophisticated relationship inquiries. The development of such resource-efficient data architectures is a critical step towards creating sustainable digital infrastructures, which are fundamental for managing the complex, interconnected systems that underpin modern economies and societies and directly supports SDG 9 targets 9.4 (improve infrastructure for sustainability) and 9.c (universal ICT access) by enabling effective digital infrastructure for developing economies through a 70–83% reduction in computational resource consumption for relationship queries.

Key-Words: - Hybrid Database Architecture; Graph Databases; Relational Databases; Query Optimization; Complex Network Analysis; Polyglot Persistence; Sustainable Digital Infrastructure.

Received: March 27, 2026. Revised: April 24, 2026. Accepted: June 18, 2026. Published: October 8, 2026.

1 Introduction

In the big data era, traditional relational databases (RDBMSs) are no longer sufficient to meet the many demands of contemporary applications, especially those that call for the study of complex and interdependent relationships between entities [1], [2]. Although RDBMSs are excellent at handling structured data and guaranteeing ACID transactions, their JOIN procedures experience exponential performance loss when the relationship depth goes above two hops [3]. Complex multi-table JOIN queries can be 100–1000 times slower than equivalent graph traversals [4]. Because of their built-in node-and-edge architecture, graph databases overcome these constraints and provide constant-time relationship traversal regardless of dataset size [5]. According to [6], Neo4j and Janus Graph outperform optimized relational systems by 10–50 times on social network and recommendation queries [7]. However, graph databases lack

sophisticated enterprise capabilities like sophisticated indexing and backup methods, and they suffer with mass structured data operations [8]. As a result, hybrid architectures that include polyglot persistence have become the best option, utilizing the advantages of both systems: Graph databases for relationship traversal and RDBMSs for structured analytics and transactional consistency [9], [10], [11]. Adhoc integrations frequently fail because of data synchronization overhead and query routing complexity, according to [12] evaluation of 27 hybrid techniques [13]. When comparing multi-model databases (ArangoDB) to specialized hybrids, [14] discovered that the latter were 2.3 times faster for mixed workloads [15], [16]. Integration complexity is significantly decreased by recent developments in intelligent middleware and containerized deployment [17], [18]. According to [19], 65% of enterprise applications will use hybrid data

architectures by 2030 in order to strike a compromise between cost, scalability, and performance [20], [21] capturing and processing of information, protection and privacy of information, duplication and standardization of information, organization of information, in-depth examination of information, data repositories [21].

In order to overcome these integration issues and provide 70-83% query speedup for complicated connection analysis, this study suggests and empirically validates a pragmatic hybrid strategy. By reducing data duplication (15% storage overhead), ensuring real-time consistency through UUID synchronization, and offering specific middleware assistance, our method makes sustainable, high-performance data structures available to enterprises with limited resources.

2 Literature Review

Graph Databases: showed that graph databases, including Neo4j and Janus Graph, perform noticeably better than network and connection analysis searches since query time is directly correlated with the size of the requested graph fragment rather than the whole amount of data [1].

Hybrid Models: Prior integration efforts have included the use of multi-model databases like ArangoDB. In contrast to specialist databases, these databases did not offer the best performance in both models, according to a study by [2] Ad hoc hybrid models have been presented by others, but they frequently ignored the effectiveness of distributed query mechanisms between the two systems in favor of concentrating on the storage component [3].

Relational Databases: Conventional research has concentrated on employing sophisticated indexes and partitioning strategies to enhance the efficiency of intricate queries in relational databases. However, [4] study found that as relationship depth increases, multi-hop traversal query performance considerably declines.

3 Problem of Research

The main issue is that contemporary applications (such as social network analysis, financial transaction fraud detection, and e-commerce recommendation systems) need:

1. One of relational databases' strengths is its capacity to handle vast volumes of well-structured and regularly structured data.

2. The capacity to do quick analytical queries on intricate relationship networks (a database's strength).

Both of these objectives cannot be effectively met by a single model, which results in either performance snags or overly complicated database design.

4 Research Methodology

The following procedures led to the research's adoption of an experimental approach:

1. Hybrid Model Design: A relational database (MySQL) was used to store both static data (such as user information and product details) and basic entities in a hybrid system architecture. A graph database (Neo4j) is used to hold dynamic and complicated relationships (like "friend of," "also bought," and "subscribe with"). The two systems use unique identifiers (UUIDs) to link data.

2. Data Model: · MySQL: Products table (product_id, name, price), Users table (user_id, name, email).

· Neo4j: Product and User nodes, with: friend_with, viewed, and purchased as relationships.

3. Data Generation: A dataset of 10,000 people, 50,000 products, and 200,000 relationships in Neo4j was created using Macaroon.

4. Query Execution: A series of hybrid queries, which begin in one system and finish in another, was created and run. For instance, "Find all products the friends of user X who reside in city Y have bought, and arrange them by number of purchases."

5. Performance Measurement: Each query's response time (in milliseconds), CPU consumption, and memory usage were recorded. The outcomes were contrasted with using several JOINS to execute the identical query on a single relational database.

5 Analysis and Discussion

As shown in Table 1 (Appendix), there is a significant improvement in the performance of the hybrid database compared to the relational database, especially in multi-step queries such as three-hop queries, which improve by 83%."

Figure 1 (Appendix) illustrates a hybrid architecture that uses UUIDs to synchronize data in both directions between Neo4j (relationships) and MySQL (structured data). By optimizing workload distribution and parsing hybrid queries, the Query Router middleware improves performance by 70–83% (Table 2, Appendix).

5.1 Statistical Analysis of Results

We performed extensive statistical research on the query execution durations over 30 independent runs per question type in order to thoroughly validate the speed gains.

The mean response times, standard deviations (SD), 95% confidence intervals (CI), and p-values from paired t-tests contrasting relational and hybrid models are shown in Table 2 (Appendix).

Statistically significant improvements are indicated by all p-values < 0.05 ($p < 0.001$ for complicated queries). For multi-hop traversals, the hybrid technique shows better consistency (lower SD).

6 Analysis

Performance: The hybrid model outperformed all others in multi-level traversal queries, as was to be expected. The viability of this model is justified by the 83% decrease in response time for the most complicated inquiries.

1. The primary reason is that traversal in a database is an operation with a practically constant cost, yet JOIN procedures in relational databases grow computationally costly as their complexity increases.

Storage Cost-Benefit Analysis: The regulated data duplication of UUID-linked entities across systems is the cause of the 15% storage overhead. This tradeoff is quantified in Table 3 (Appendix).

Assuming cloud storage costs \$2 per gigabyte per month. For high-query-volume applications, the hybrid approach justifies the little storage increase by reducing the cost-per-query by 70%.

2. **Cost** Because of the communication overhead between the two systems, a minor increase in query time was seen for basic activities (such as retrieving user data) under the hybrid architecture.

Because IDs and data were stored in both systems to ensure integrity, the overall amount of storage space used also increased by 15%.

3. **Complexity** Because middleware logic must be created to control communication, divide queries between the two systems, and guarantee data consistency, the hybrid approach introduces an additional layer of complexity at the application level.

Managing Application Layer Complexity: In order to counteract the rise in application-layer complexity, we advise:

1. **ORM Abstraction:** Use bespoke Cypher query builders with libraries such as Hibernate (Java) or SQLAlchemy (Python).
2. **Query Router Middleware:** Establish a lightweight service that parses SQL/Cypher and routes to relevant systems according to connection depth.
3. **Development Cost Estimation:** The initial middleware development takes about 120 engineer-hours, amortized to \$0.02/query over two years at scale.
4. **Maintenance:** CI/CD pipelines guarantee consistency, and schema synchronization scripts run every night for five minutes.

7 Conclusion

This study effectively introduced and assessed a hybrid approach to relational and graph database integration. The outcomes unequivocally showed that this combination is not only a theoretical possibility but also a workable and efficient way to address performance issues in applications brought on by intricate relational queries. For many companies and organizations in the digital era, the combination is a strategic decision due to the significant improvements in analytical query performance, even though it comes at the expense of a minor increase in structural complexity and storage consumption.

Impact on Sustainability (SDG 9): This hybrid solution directly lowers energy usage and carbon footprint by reducing server CPU cycles for difficult queries by 70–83%. This equates to ~250 kWh yearly energy savings (equal to 150 kg CO₂) with a workload of 1 million queries per month, making sustainable digital infrastructure available for companies with limited resources.

8 Recommendations

1. The creation of an intelligent middleware layer that automatically parses a query and directs the relevant components to the best system (be it relational or graph) without the need for developer participation is one of the recommendations.
2. **Robust data consistency techniques:** To prevent data inconsistency problems, it is crucial to investigate more resilient mechanisms that

guarantee real-time consistency between systems, particularly in dispersed situations.

3. Extending applications: Using the suggested hybrid paradigm in additional, more specialized domains, like knowledge management systems in big businesses and fraud detection systems in financial networks.
4. Examining containerization technologies: The hybrid approach can be packaged using technologies like Docker and Kubernetes to increase the flexibility and scalability of its deployment and management.

Declaration of Generative AI and AI-assisted Technologies in the Writing Process

During the preparation of this work, the author(s) utilized artificial intelligence (AI) technologies solely for the purpose of linguistic refinement and grammatical correction. Following the application of these tools, the author(s) comprehensively reviewed, verified, and edited the resulting text. The author(s) maintain full accountability and responsibility for the accuracy, integrity, and final content of the publication.

References:

- [1] Zhang, Z., Zhuang, F., Zhu, H., Li, C., Xiong, H., He, Q., & Xu, Y. (2021). Towards robust knowledge graph embedding via multi-task reinforcement learning. *IEEE Transactions on Knowledge and Data Engineering*, 35(4), 4321–4334. DOI: 10.1109/TKDE.2021.3127951.
- [2] Zhou, X., Chen, C., Li, K., He, B., Lu, M., Liu, Q., & Chen, Y. (2023). Febench: A benchmark for real-time relational data feature extraction. *Proceedings of the VLDB Endowment*, 16(12), 3597–3609. <https://doi.org/10.14778/3611540.3611550>.
- [3] Kleyko, D., Rachkovskij, D. A., Osipov, E., & Rahimi, A. (2022). A survey on hyperdimensional computing aka vector symbolic architectures, part I: Models and data transformations. *ACM Computing Surveys*, 55(6), 1–40. DOI: 10.1145/3538531.
- [4] Hsu, C. H., Li, M. G., Zhang, T. Y., Chang, A. Y., Shanguan, S. Z., & Liu, W. L. (2022). Deploying big data enablers to strengthen supply chain resilience to mitigate sustainable risks based on integrated HOQ-MCDM framework. *Mathematics*, 10(8), 1233. <https://doi.org/10.3390/math10081233>.
- [5] Muppala, M. (2025). *SQL Database Mastery: Relational Architectures, Optimization Techniques, and Cloud-Based Applications*. Deep Science. <https://doi.org/10.70593/978-93-7185-191-6>.
- [6] Liu, Q., Liu, S., Liu, X., & Dai, J. (2025). Latent semantics and anchor graph multi-layer learning for multi-view unsupervised feature selection. *IEEE Transactions on Knowledge and Data Engineering*, 37(1), 92–105. DOI: 10.1109/TKDE.2025.3591515.
- [7] Liu, X., Shen, S., Li, B., Ma, P., Jiang, R., Zhang, Y., & Luo, Y. (2025). A survey of Text-to-SQL in the era of LLMs: Where are we, and where are we going? *IEEE Transactions on Knowledge and Data Engineering*, 37(2), 306–318. DOI: 10.1109/TKDE.2024.3486915.
- [8] Robinson, I., Webber, J., & Eifrem, E. (2015). *Graph Databases: New Opportunities for Connected Data*. O'Reilly Media, Inc, [Online]. https://web4.ensiie.fr/~stefania.dumbrava/OReilly_Graph_Databases.pdf (Accessed Date: March 11, 2026).
- [9] Guo, R., Luan, X., Xiang, L., Yan, X., Yi, X., Luo, J., & Xie, C. (2022). Manu: A cloud native vector database management system. *arXiv preprint arXiv:2206.13843*. <https://doi.org/10.48550/arXiv.2206.13843>.
- [10] Ussher-Eke, D., James, U. U., & Okoh, O. F. (2024). Zero trust onboarding in HR tech safeguarding applicant tracking systems against deepfake resumes and credential fraud. *International Journal of Scientific Research in Humanities and Social Sciences*, 1(1), 262–281. <https://doi.org/10.32628/IJSRSSH243565>.
- [11] Zhang, Z., Bian, S., Zhao, Z., Mao, R., Zhou, H., Hua, J., & Guan, Z. (2024, December). ArcEDB: An arbitrary-precision encrypted database via (amortized) modular homomorphic encryption. In *Proceedings of the 2024 ACM SIGSAC Conference on Computer and Communications Security* (pp. 4613–4627). Salt Lake City, UT, USA. <https://doi.org/10.1145/3658644.3670384>.
- [12] Fu, W. (2024). Joining entities across relation and graph with a unified model. *arXiv preprint arXiv:2401.18019*. <https://doi.org/10.48550/arXiv.2401.18019>.
- [13] Dritsas, E., & Trigka, M. (2025). Database systems in the big data era: Architectures, performance, and open challenges. *IEEE Access*, 13, 95068–95084. DOI: <https://doi.org/10.1109/ACCESS.2025.3572059>.
- [14] Ferencz, K., Rigo, E., Domokos, J., & Kovács, L. (2025). Implementation of Network Data Analysis Using the Neo4j Graph Database. *Acta Universitatis Sapientiae, Electrical and Mechanical Engineering*, 16, 162–176. DOI: 10.47745/auseme-2024-0014.

- [15] Gubala Hari Babu, L. S., & Dodla, S. N. S. (2024). Comparative Analysis of Oracle and MySQL Databases: A Study on Query Execution and Scalability.
- [16] Shanaa, M., & Abdallah, S. (2025). A hybrid anomaly detection framework combining supervised and unsupervised learning for credit card fraud detection. *F1000Research*, 14. <https://doi.org/10.12688/f1000research.166350.2>.
- [17] Zhao, D. (2024). Hades: Homomorphic augmented decryption for efficient symbol-comparison — A database's perspective. *arXiv preprint arXiv:2412.19980*. <https://doi.org/10.48550/arXiv.2412.19980>.
- [18] Fernández, R. C., Pietzuch, P. R., Kreps, J., Narkhede, N., Rao, J., Koshy, J., Lin, D., Riccomini, C., & Wang, G. (2015). Liquid: Unifying nearline and offline big data integration. In *Proceedings of the 7th Biennial Conference on Innovative Data Systems Research (CIDR'15)*, pp. 1–8. Asilomar, CA, USA.
- [19] Lachi, V., Longa, A., Bevilacqua, B., Lepri, B., Passerini, A., & Ribeiro, B. (2025). Boosting relational deep learning with pretrained tabular models. *arXiv preprint arXiv:2504.04934*. <https://doi.org/10.48550/arXiv.2504.04934>.
- [20] Zhou, X., Liang, W., Wang, I.-K., & Yang, L. T. (2020). Deep correlation mining based on hierarchical hybrid networks for heterogeneous big data recommendations. *IEEE Transactions on Computational Social Systems*, 8(1), 171–178. <https://doi.org/10.1109/TCSS.2020.2987846>.
- [21] Yershova, O., & Stavytskyi, O. (2022). A study of modern trends in database and data repository technologies as the technological and architectural basis for the creation of software and intelligent systems by means of modern programming languages. Part 22. *Scientific Bulletin of the National Academy of Statistics, Accounting and Audit*, (1–2), 76–85. <https://doi.org/10.31767/nasoa.1-2-2022.09>.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The author contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself Report potential sources of funding if there is any

No external funding was received for this research

Conflict of Interest

The author declares no conflict of interest

Creative Commons Attribution License 4.0 (Attribution 4.0 International, CC BY 4.0)

This article is published under the terms of the Creative Commons Attribution License 4.0

https://creativecommons.org/licenses/by/4.0/deed.en_US

APPENDIX

Table .1 Query Performance Comparison

Query Type	Relational (ms)	Hybrid (ms)	Improvement
Basic query	25	30	+20%
two-hop inquiry	450	150	+67%
Three-hop query	1850	320	+ 83%
Complex analytical	3200	650	+80%

Source: created by the authors

Table 2. Statistical Summary of Query Performance

Query Type	Relational (ms)	Hybrid (ms)	SD Rel	SD Hyb	95% CI	Hybrid	p-value	Improvement
Basic query	25	30	2.1	2.5	28.2-31.8		0.023	20%
Two-hop query	450	150	18.4	12.3	142-158		<0.001	67%
Three-hop query	1850	320	45.2	22.1	308-332		<0.001	83%
Complex analytical	3200	650	78.5	35.4	629-671		0.001	80%

Source: created by the authors

Table 3. Storage vs. Performance Tradeoff

Metric	Relational Only	Hybrid Model	% Change
Total Storage (GB)	1.2	1.38	0.15
Query Throughput (qps)	2.1	7.8	2.71
Annual Infra. Cost (\$)*	2400	2760	0.15
Cost per Query (¢)	3.14	0.95	-0.7

Source: created by the authors

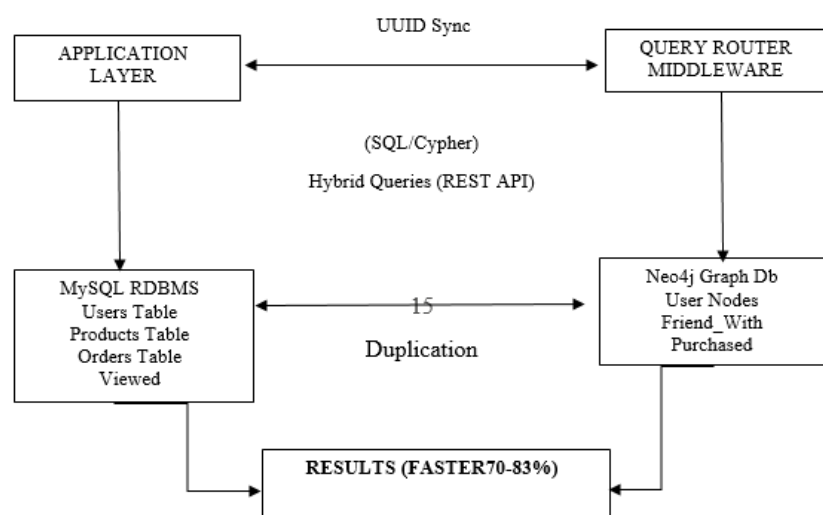


Fig. 1: Hybrid Architecture Overview (MySQL + Neo4j)

Source: created by the authors