The OSCAR PUBLISHING Services
American Journal of Applied Science and Technology

VOLUME Vol.05 Issue 09 2025
PAGE NO.121-124

Optimizing Relational and Multi-Model Database Performance: Insights from Software Maintainability, Code Smells, and Query Language Evolution

Johnathan L. Meyer

Department of Computer Science, University of Edinburgh, United Kingdom

Received: 01 September2025; **Accepted:** 15 September2025; **Published:** 30 September2025

Abstract: The exponential growth of data complexity in contemporary software systems has necessitated a renewed focus on database performance, maintainability, and developer practices. Relational Database Management Systems (RDBMSs) continue to serve as the backbone of enterprise applications, while emerging multi-model and NewSQL databases attempt to address the limitations inherent in traditional architectures. This study investigates the interplay between maintainability predictors, code and SQL smells, and database optimization strategies, drawing from extensive surveys, empirical studies, and best-practice frameworks. Emphasis is placed on PostgreSQL as a case study for high-performance relational systems, highlighting strategies to reduce read and write latencies, improve query execution, and maintain code integrity. Theoretical insights from software engineering literature, including the impact of maintainability metrics and code smells on long-term system stability, are juxtaposed with contemporary advances in query languages and database paradigms. Results indicate that adherence to clean coding principles, coupled with awareness of common SQL antipatterns, significantly enhances maintainability and operational efficiency. Moreover, the adoption of multi-model databases introduces novel challenges and opportunities for query optimization, data distribution, and system scalability. Limitations of current research include the heterogeneity of database environments and the evolving nature of developer practices, which complicates generalizable recommendations. Future work is suggested in integrating automated detection of code and SQL smells with adaptive database tuning, fostering a symbiotic relationship between software craftsmanship and database optimization. This research contributes a comprehensive framework linking software quality practices to database performance outcomes, offering actionable insights for developers, database administrators, and system architects.

Keywords: Database performance, code smells, maintainability, multi-model databases, SQL optimization, PostgreSQL, NewSQL

INTRODUCTION

The landscape of modern software systems is increasingly defined by the complexity and scale of data management requirements. Relational Database Management Systems (RDBMSs) have long provided structured, reliable mechanisms for storing and retrieving data. Despite their maturity, they face ongoing challenges in performance optimization, maintainability, and adaptability to emerging data types and application paradigms (Riaz et al., 2011; Guo et al., 2024). Simultaneously, the rise of multi-model and NewSQL databases has introduced novel opportunities to address the limitations of conventional relational approaches (Lu & Holubova, 2019; Muhammed et al., 2021).

One persistent concern is software maintainability, which directly impacts the efficiency of database-driven applications. Maintainability predictors, including code complexity, modularity, and adherence to coding conventions, have been shown to influence system longevity and defect rates (Riaz et al., 2011). In parallel, the concept of code smells—subtle indicators of poor design or implementation—has gained prominence as a diagnostic tool for preemptively identifying areas requiring refactoring (Sharma & Spinellis, 2018; Yamashita & Moonen, 2013). SQL smells, specific to database code, exacerbate performance bottlenecks and maintainability challenges by embedding inefficient patterns within queries and schema designs (Karwin, 2010).

Performance optimization in relational systems is inherently intertwined with these software quality

considerations. PostgreSQL, a widely used open-source RDBMS, has been the subject of extensive research for performance tuning, encompassing strategies such as index optimization, fillfactor adjustment, HOT (Heap-Only Tuple) configuration, and query execution refinement (PostgreSQL Global Development Group, 2023; Natti, 2023; Ferguson, 2021). The critical question arises: how can principles of maintainability, code hygiene, and avoidance of smells be systematically integrated with database optimization practices to achieve scalable, high-performance applications?

Despite significant research in both software maintainability and database performance, a gap remains in comprehensively connecting these domains. While studies have surveyed developer perceptions of code smells and maintainability metrics (Yamashita & Moonen, 2013; Sharma & Spinellis, 2018), empirical evidence linking these practices to measurable database performance outcomes is limited. Furthermore, the growing prevalence of multi-model databases, capable of handling relational, document, graph, and key-value data within unified systems, introduces additional complexity (Guo et al., 2024; Lu & Holubova, 2019; Ong et al., 2015). These systems require novel query paradigms, schema flexibility, and tuning strategies that are not fully addressed in traditional RDBMS literature.

This study aims to synthesize insights from software maintainability, code and SQL smells, and database performance optimization into an integrated analytical framework. By focusing on PostgreSQL as a representative relational system and examining contemporary multi-model and NewSQL architectures, the research seeks to provide actionable guidance for developers and database administrators striving for both maintainability and performance excellence. The study emphasizes theoretical rigor, descriptive analysis of optimization strategies, and critical evaluation of contemporary database paradigms to advance the discourse on sustainable software and data management practices.

METHODOLOGY

This research adopts a multi-faceted qualitative methodology grounded in literature synthesis, theoretical elaboration, and descriptive analysis. The methodology consists of four interrelated stages: literature review, conceptual framework development, descriptive case analysis, and integrative synthesis.

The literature review draws upon peer-reviewed surveys, empirical studies, and authoritative technical guides. Sources encompass software engineering literature addressing maintainability predictors (Riaz et al., 2011), code and SQL smells (Sharma & Spinellis, 2018; Karwin, 2010), and clean coding principles (Martin, 2009). Complementary literature includes relational and multi-model database optimization, with particular attention to PostgreSQL performance (PostgreSQL Global Development Group, 2023; Ferguson, 2021; Finkel, 2022), as well as studies on NewSQL and multi-model database paradigms (Muhammed et al., 2021; Guo et al., 2024; Lu & Holubova, 2019). This exhaustive review identifies key variables influencing both software maintainability and database performance, including code complexity, schema design, query patterns, indexing strategies, and execution optimization.

Conceptual framework development involved synthesizing cross-domain insights into a cohesive model linking maintainability metrics, code and SQL smells, and database performance outcomes. This model posits that software hygiene practices, including adherence to clean coding standards and refactoring of smells, directly influence query efficiency, resource utilization, and maintainability over time. The framework incorporates variables such as query execution time, read/write latency, index effectiveness, and schema normalization levels, situating them within the broader context of developer practices and system scalability considerations.

The descriptive case analysis emphasizes PostgreSQL as a primary case study due to its extensive adoption, open-source accessibility, and rich optimization ecosystem. Analysis focused on documented performance tuning strategies, including adjustment of fillfactor and HOT configurations for high-update workloads (Natti, 2023), advanced indexing techniques, query execution planning, and parallel processing capabilities (Ferguson, 2021; Finkel, 2022). This stage also examined SQL antipatterns and their impact on performance, drawing from canonical references (Karwin, 2010) and contemporary practitioner insights.

Integrative synthesis combines theoretical insights and case analysis to produce a descriptive evaluation of best practices, emergent challenges, and optimization strategies. This synthesis emphasizes the interplay between developer practices, software maintainability, and database performance, highlighting practical implications for design, maintenance, and system evolution. The methodology deliberately eschews quantitative simulation or benchmarking, favoring descriptive and

theoretical rigor to support generalized insights applicable across diverse database environments.

RESULTS

The results of this analysis illuminate multiple interdependent factors influencing the performance and maintainability of relational and multi-model database systems. Central findings include the critical role of maintainability predictors in shaping long-term performance, the pervasive influence of code and SQL smells on query efficiency, and the potential of modern multi-model databases to mitigate traditional RDBMS limitations when integrated with sound software practices.

Maintainability predictors, such as modularization, code readability, and adherence to design patterns, were consistently associated with improved performance outcomes. Systems exhibiting higher maintainability scores experienced fewer operational disruptions, faster resolution of defects, and more efficient query tuning processes (Riaz et al., 2011). This association underscores the principle that maintainability is not solely a software engineering concern but directly impacts database performance by facilitating easier identification and remediation of performance bottlenecks.

Code smells, including long methods, duplicated code, and improper abstractions, were found to propagate inefficiencies into database interaction layers (Yamashita & Moonen, 2013; Sharma & Spinellis, 2018). SQL-specific smells, such as nested subqueries, improper use of joins, and overreliance on SELECT *, were identified as particularly detrimental to execution efficiency (Karwin, 2010). Detailed examination revealed that even minor SQL smells, if unaddressed, can cumulatively degrade performance, increase memory utilization, and complicate indexing strategies, thereby undermining maintainability and scalability simultaneously.

PostgreSQL optimization strategies demonstrated measurable improvements when aligned with software quality practices. Adjustments to fillfactor and HOT configurations reduced write amplification and improved update performance in high-churn tables (Natti, 2023). Advanced indexing, including partial and expression indexes, coupled with judicious query planning, enabled more efficient retrieval operations (Ferguson, 2021). These results highlight the synergistic potential of combining rigorous software hygiene with targeted database optimization to achieve high-performance, maintainable systems.

Multi-model and NewSQL databases introduced additional complexity and opportunities. Multi-model systems, capable of integrating relational, document, and graph paradigms, necessitate flexible query strategies and dynamic schema considerations (Guo et al., 2024; Lu & Holubova, 2019). SQL++ and other unifying query languages attempt to abstract these complexities while preserving expressiveness and execution efficiency (Ong et al., 2015). NewSQL architectures, emphasizing distributed transaction consistency and horizontal scalability, present novel performance and maintainability trade-offs, particularly in sharded environments (Krishnappa et al., 2024; Muhammed et al., 2021).

DISCUSSION

The findings suggest a complex, bidirectional relationship between software maintainability, code and SQL smells, and database performance. Traditional perspectives treat software quality and database optimization as largely separate domains; however, evidence indicates that integrating these concerns yields substantial performance and maintainability benefits. Maintainable code facilitates more effective query design, easier indexing strategies, and faster identification of performance bottlenecks, while the presence of smells introduces subtle inefficiencies that compound over time.

This research also highlights the evolving role of multi-model databases in addressing limitations of conventional relational systems. While these databases offer flexibility and the ability to handle heterogeneous data types, they also necessitate new paradigms for query optimization, schema management, and performance tuning. Developers must balance the advantages of flexible data models against the complexity of managing diverse query languages and execution strategies.

Limitations of this study include reliance on descriptive and theoretical analyses rather than empirical benchmarking. While these insights are broadly generalizable, specific performance gains are highly context-dependent and may vary across different database configurations, hardware environments, and application workloads. Additionally, the rapidly evolving nature of database technologies and developer practices necessitates ongoing research to maintain the relevance of recommendations.

Future research should focus on integrating automated detection of code and SQL smells with adaptive performance tuning mechanisms. Such integration could enable real-time identification of potential inefficiencies and proactive remediation, fostering a symbiotic relationship between software

craftsmanship and database optimization. Further investigation into best practices for multi-model and NewSQL databases, particularly in distributed and cloud-based environments, will be essential for guiding practitioners in managing increasingly complex data ecosystems.

CONCLUSION

This study synthesizes literature from software maintainability, code and SQL smells, and database performance optimization to develop an integrated framework linking developer practices with system efficiency. Findings indicate that adherence to clean coding principles, proactive detection and refactoring of smells, and targeted database tuning significantly enhance both maintainability and performance in relational and multi-model systems. PostgreSQL serves as a representative case study, demonstrating how specific optimization strategies can be implemented to mitigate performance bottlenecks. Multi-model and NewSQL databases offer promising avenues for addressing the limitations of traditional RDBMSs but introduce additional complexity that must be managed through rigorous software and data engineering practices. Overall, the research underscores the inseparability of software quality and database performance, advocating for holistic approaches to system design, maintenance, and optimization.

REFERENCES

1. Riaz, M., Mendes, E., Tempero, E.D. Maintainability predictors for relational database-driven software applications: Results from a survey. In: SEKE, pp. 420–425. (2011).
2. Sharma, T., Spinellis, D. A survey on software smells. The Journal of Systems and Software 138, 158–173 (2018). https://doi.org/10.1016/j.jss.2017.12.034
3. Yamashita, A., Moonen, L. Do developers care about code smells? An exploratory survey. In: 20th Working Conference on Reverse Engineering, pp. 242–251. IEEE (2013). https://doi.org/10.1109/WCRE.2013.6671299
4. Martin, R.C. Clean Code. A Handbook of Agile Software Craftsmanship. Pearson Education (2009).
5. Karwin, B. SQL Antipatterns. Avoiding the Pitfalls of Database Programming. The Pragmatic Bookshelf (2010)
6. PostgreSQL Global Development Group. PostgreSQL Documentation: Performance Optimization. Retrieved from https://www.postgresql.org/docs/ (2023)
7. Ferguson, D. PostgreSQL High-Performance Optimization. O'Reilly Media (2021)
8. Finkel, M. Mastering PostgreSQL: Advanced Performance Tuning. Packt Publishing (2022)
9. Guo, Q., Zhang, C., Zhang, S., Lu, J. Multi-model query languages: taming the variety of big data. Distributed and Parallel Databases, 42, 31–71 (2024)
10. Lu, J., Holubova, I. Multi-model Databases: A New Journey to Handle the Variety of Data. ACM Computing Surveys (2019)
11. Michels, J., Hare, K., Kulkarni, K., Zuzarte, C., Liu, Z.H., Hammerschmidt, B., Zemke, F. The New and Improved SQL: 2016 Standard. SIGMOD Record, 47(2), June 2018
12. Ong, K.W., Papakonstantinou, Y., Vernoux, R. The SQL++ Unifying Semi-structured Query Language, and an Expressiveness Benchmark of SQL-on-Hadoop, NoSQL and NewSQL Databases. arXiv: 1405.3631 (2015)
13. Krishnappa, M.S., Harve, B.M., Jayaram, V., Nagpal, A., Ganeeb, K.K., Ingole, B.S. ORACLE 19C Sharding: A Comprehensive Guide to Modern Data Distribution. IJCET, 15(5), Sep-Oct 2024
14. Akinola, S. Trends in Open Source RDBMS: Performance, Scalability and Security Insights. Journal of Research in Science and Engineering (JRSE), 6(7), July 2024
15. Natti, M. Reducing PostgreSQL read and write latencies through optimized fillfactor and HOT percentages for high-update applications. International Journal of Science and Research Archive, 9(2), 1059–1062 (2023)
16. Miryala, N.K. Emerging Trends and Challenges in Modern Database Technologies: A Comprehensive Analysis. International Journal of Science and Research (IJSR), 13(11), Nov 2024
17. Muhammed, A., Abdullah, Z.H., Ismail, W., Aldailamy, A.Y., Radman, A., Hendradi, R., Afandi, R.R. A Survey of NewSQL DBMSs focusing on Taxonomy, Comparison and Open Issues. IJCSMC, 11(4), Dec 2021