The OSCAR PUBLISHING Services
American Journal of Applied Science and Technology

VOLUME Vol.05 Issue 10 2025
PAGE NO.297-302

Towards an Integrated Framework for Database Schema Quality, Refactoring, and Performance Optimization in Modern Relational and NewSQL Systems

Musayeva Adiba Abdumajidovna
Department of Computer Science, Global University

Received: 01 October 2025; **Accepted:** 15 October 2025; **Published:** 31 October 2025

Abstract: Database-driven applications form the backbone of modern information systems, spanning domains from enterprise resource planning to big data analytics. As database systems evolve, both the logical design of schemas and the physical performance tuning become critical — yet these two aspects are often addressed in isolation. This paper proposes an integrated conceptual framework that unifies database schema quality assessment, refactoring practices, and performance optimization strategies in contemporary Relational (RDBMS) and NewSQL systems. Drawing on a comprehensive review of existing literature on database anti-pattern detection, schema refactoring, technical debt in schemas, and performance tuning techniques, we systematically analyze how schema "smells," technical debt, and performance bottlenecks interplay. We articulate a taxonomy of schema issues (structural flaws, anti-patterns, performance-influencing design choices) and map corresponding refactoring/migration patterns, optimization methods, and evaluation mechanisms. Our framework emphasizes continuous schema evolution in response to changing application workload and data characteristics. We discuss the theoretical implications for maintainability, performance, and adaptability, and provide guidelines for practitioners on prioritizing refactorings and performance enhancements. We conclude with limitations of the conceptual study and outline future research directions, including empirical validation, tool support, and integration into DevOps pipelines.

Keywords: Database schema quality; Schema refactoring; Technical debt; NewSQL; PostgreSQL performance; Database anti-patterns

INTRODUCTION

Relational databases remain dominant in enterprise applications due to their expressive schema definition capabilities, ACID guarantees, and mature ecosystem. However, as applications evolve — new features added, data volume grows, usage patterns shift — the original schema may begin to exhibit structural deficiencies, performance bottlenecks, or maintainability issues. These problems often arise gradually, accumulating "technical debt" in the schema that complicates future changes, degrades performance, and undermines reliability. Meanwhile, performance optimization might tempt database administrators (DBAs) to apply ad-hoc tuning (e.g., fill-factor adjustments, partitioning, indexing, configuration tweaks) without addressing the underlying schema design flaws. In modern data-intensive applications, especially under high transaction loads or big data workloads, this disconnect between logical schema design and physical performance tuning can lead to suboptimal and fragile systems.

Over the past two decades, research on database schema refactoring, anti-pattern detection, and schema quality measurement has gained renewed interest. Seminal works include literature on industrial reverse engineering of databases (Blaha, 2001), catalogs of database refactorings, and empirical studies on "smelly relations" (Sharma et al., 2018), anti-patterns detection (Khumnin & Senivongse, 2017), and using query-based methods to surface design flaws (Eessaar, 2015; Eessaar & Voronova, 2015). Concurrently, advances in performance tuning, especially for modern systems like PostgreSQL and NewSQL platforms, address latency, concurrency, partitioning, and scalability (Momjian, 2020; SQL Performance Blog, 2021; Petkov, 2023; Natti, 2023; Maia, 2015; Moniruzzaman, 2014). Yet, there remains a gap: a cohesive framework that bridges schema-quality practices and performance optimization, guiding practitioners in evolving schema responsibly while sustaining performance.

Such integration matters, because schema refactorings and physical optimizations are not

independent: refactoring can remove structural inefficiencies that cause poor query plans, reduce redundant joins, or enable better partitioning; conversely, performance-oriented changes (e.g., partitioning, sharding) may introduce structural complexity or divergence from normalized schema. Ignoring this interplay can result in either wasted optimization effort, or stable but rigid schemas that hinder future adaptation.

This paper addresses this gap by building an integrated conceptual framework that unites database schema quality assessment, refactoring practices, and performance optimization strategies, applicable to both traditional RDBMS and modern NewSQL / distributed systems. By synthesizing prior works and distilling key patterns, we aim to offer both theoretical coherence and practical guidance.

METHODOLOGY

This research adopts a structured literature-synthesis and conceptual modeling approach. We collected and analyzed peer-reviewed conference proceedings, journal articles, technical reports, and practitioner blogs that address database schema design quality, refactoring practices, anti-pattern detection, performance tuning, and modern scalable database solutions. Key sources span from foundational works (e.g., schema modeling foundations, multi-paradigm data integration), through reverse-engineering and refactoring efforts, to recent advances in performance-oriented database systems.

Our method consisted of the following phases:

1. **Literature Collection and Categorization**: Using the provided references as seed documents, we identified related works in schema quality (e.g., anti-pattern detection, schema smells, refactoring catalogs), performance optimization (e.g., PostgreSQL tuning, partitioning, concurrency control), and NewSQL/NoSQL/multi-model systems research.

2. **Qualitative Thematic Analysis**: Through iterative reading, we coded recurring themes: types of schema defects (e.g., normalization violations, naming inconsistencies), anti-patterns, schema refactorings, detection methods (query-based, static analysis), technical debt in schemas, performance concerns (latency, concurrency, partitioning), and scalability.

3. **Taxonomy Construction:** We organized themes into a hierarchical taxonomy delineating schema-level issues, performance-level issues, and the overlap (e.g., design choices that influence performance).

4. **Framework Design**: Based on the taxonomy, we designed an integrated framework that maps problematic patterns to remediation strategies (refactoring, optimization, monitoring) and evaluation mechanisms (static analysis, workload-based testing, performance metrics).

5.**Conceptual Validation via Argumentation:** We critically discuss how existing refactoring and performance techniques align with, or fall short of, the needs articulated by the framework. We examine trade-offs, limitations, and potential conflicts when combining refactoring and optimization.

Because the scope is conceptual rather than empirical, we do not conduct real-world database modifications; rather, we rely on prior empirical findings, theoretical reasoning, and practitioner insights. This approach allows us to propose generalizable patterns and guidelines, while acknowledging the need for future empirical validation.

RESULTS

Our literature synthesis and analysis produced a set of core findings, culminating in a taxonomy of schema and performance issues, a mapping to remediation strategies, and a conceptual model — herein referred to as the Integrated Schema–Performance Evolution (ISPE) Framework.

Taxonomy of Issues

Schema-Quality Issues (“Logical-Level”)

- **Normalization Violations and Redundancy**: Instances where schema design violates normal forms, leading to data redundancy, update anomalies, and maintenance overhead. These often persist when databases age without regular refactoring.
- **Naming and Semantic Inconsistencies**: Poor naming conventions, ambiguous attribute names, inconsistent data types or lengths across similar entities — symptomatic of ad-hoc schema evolution.
- **Structural Anti-Patterns / Smells**: High coupling between tables, overly wide relations, excessive nullability, presence of rarely used columns — these undermine maintainability and comprehension (Sharma et al., 2018; Khumnin & Senivongse, 2017).
- **Hidden Technical Debt:** Deferred schema improvements, temporary fixes that become permanent, lack of documentation — long-term impediments to scalability and maintainability (Weber et al., 2014; Vial, 2015).

Performance-Level Issues (“Physical-Level” or “Operational”)

- **Inefficient Query Plans**: Poor schema design leading to suboptimal joins or indexing, resulting in high query latency or resource consumption (Momjian, 2020; Petkov, 2023).
- **Concurrency Bottlenecks and Locking/Isolation Inefficiencies:** High contention under heavy transactional workloads; suboptimal isolation configurations causing blocking or performance degradation (Cahill, 2009; Momjian, 2020).
- **Inadequate Partitioning or Sharding:** Lack of partitioning or poorly chosen partitioning schemes causing uneven data distribution, slow queries, and scaling problems (Maia, 2015; Moniruzzaman, 2014).
- **Suboptimal Physical Storage Configurations:** Improper fill-factor, lack of HOT (Heap Only Tuple) tuning, inefficient vacuum/autovacuum settings — all contributing to write amplification, bloat, and slower updates (Natti, 2023; SQL Performance Blog, 2021).

Intersecting Issues (Schema-Design ↔ Performance)

- **Schema Designs Inhibiting Partitioning or Sharding:** Deeply normalized or overly complex relational models that resist simple partitioning/sharding strategies.
- **Schema Smells Leading to Performance Degradation:** Wide relations or unused columns causing larger row sizes, poor cache utilization, slower retrieval.
- **Ad-hoc Optimizations Masking Underlying Defects:** E.g., adding indexes to compensate for poor schema design, which may mitigate a performance issue but exacerbate maintenance cost and technical debt.

Integrated Schema–Performance Evolution (ISPE) Framework

The ISPE Framework proposes a continuous, cyclical process that brings together schema quality assessment, refactoring/migration, performance optimization, and monitoring. Its core components are:

1. **Schema and Performance Audit:** Using static analysis (anti-pattern detection, schema-linting) and query-based search for design flaws (Eessaar, 2015; Eessaar & Voronova, 2015), along with workload-based performance profiling (e.g., slow-query logs, latency metrics).

2. **Issue Classification and Prioritization:** Classify identified issues according to the taxonomy (logical-level vs performance-level vs intersecting). Prioritize based on severity, risk, and expected benefit (e.g., structural anti-patterns affecting many modules may be high priority; a slow but rarely used query low priority).

3. Remediation Strategy Selection: For logical issues — apply schema refactorings described in catalogs of database refactorings (Catalog of Database Refactorings), guided by experiences from practitioners (Vial, 2015). For performance issues — apply optimization techniques such as partitioning/sharding (Maia, 2015; Moniruzzaman, 2014), storage tuning (fill-factor, HOT tuning) (Natti, 2023), indexing, and configuration adjustments (SQL Performance Blog, 2021; Petkov, 2023). For intersecting issues — sometimes a hybrid approach: e.g., normalize or refactor schema, then re-partition; or denormalize selectively to enable sharding or performance.

4. Testing and Validation: After changes, run functional tests, regression tests, and performance benchmarks to ensure correctness, maintainability, and acceptable performance under realistic workloads.

5. Continuous Monitoring and Iteration: Monitor schema evolution, workload changes, performance metrics; repeat audit and remediation cycles as necessary — promoting agile schema evolution rather than static "set-and-forget."

Through this framework, a database system can evolve gracefully, preserving both structural clarity and operational efficiency.

DISCUSSION

The conceptual synthesis and formulation of the ISPE Framework reveal several important theoretical and practical implications, as well as inherent tensions and limitations.

First, the benefit of integrating schema quality and performance optimization. Historically, database research and practice have often treated schema design and performance tuning as separate concerns. Schema designers focus on normalization, semantics, and maintainability; DBAs focus on indexing, partitioning, concurrency, and hardware performance. The ISPE Framework bridges this divide: by using unified audits and classification, practitioners can co-ordinate schema refactoring and performance optimization in a single workflow. This integrated view helps avoid fragmentation, redundant efforts (e.g., ad-hoc indexing that masks poor design), and technical debt accrual. It also aligns with agile and DevOps philosophies of incremental, continuous improvement.

Second, the role of anti-pattern detection and schema smells in proactive maintenance. Works like

"Smelly Relations" (Sharma et al., 2018) and anti-pattern detection (Khumnin & Senivongse, 2017) demonstrate that many schema design issues — which may not immediately manifest as query failures — slowly degrade maintainability and adaptability. By incorporating static analysis tools and query-based detection (Eessaar, 2015; Eessaar & Voronova, 2015) into the audit phase, organizations can proactively identify "silent" defects before they cause critical issues. This proactive approach counters the common pattern of "refactor-when-painful," reducing future costs and systemic fragility.

Third, the challenges in combining refactoring with performance optimizations. While the dual lens of schema quality and performance is beneficial, it also introduces potential conflicts. For example, normalizing data to eliminate redundancy and improve maintainability may fragment data across many tables, potentially worsening performance for join-intensive queries. Conversely, denormalizing or introducing redundant attributes to optimize performance may degrade schema clarity and maintainability. Similarly, partitioning or sharding — though beneficial for scalability — may impose constraints on refactoring, or require duplication of refactoring effort across partitions. These trade-offs underline the importance of careful prioritization and testing in the remediation strategy. Without systematic validation (functional + performance tests), refactoring may introduce latent defects or degrade performance unexpectedly.

Fourth, applicability to modern NewSQL and distributed databases. The emergence of NewSQL systems — scalable, distributed, and often supporting relational semantics — complicates the traditional schema design-performance trade-off. On one hand, NewSQL platforms aim to offer both scalability and relational consistency (Moniruzzaman, 2014; Pavlo & Aslett, 2016). On the other hand, distributed execution environments amplify sensitivity to schema design: sharding strategies, data distribution, partitioning keys, replication, and concurrency control all interplay with schema structure. The ISPE Framework, being agnostic to specific RDBMS or NewSQL platforms, can guide schema evolution even in distributed contexts: by evaluating how schema changes affect data distribution, query patterns, and consistency mechanisms.

Fifth, the need for tooling and automated support. While the conceptual framework provides a valuable roadmap, realizing it in practice requires tool support: static schema analyzers, query-based design flaw detectors, refactoring automation, performance monitoring dashboards, and regression/performance test harnesses. Some progress exists — catalogues of schema refactorings, anti-pattern detection tools (Khumnin & Senivongse, 2017), and query-based detection methods (Eessaar & Voronova, 2015) — but a fully integrated toolchain that combines logical analysis, performance profiling, and automated refactoring remains lacking. Building such tooling would significantly reduce the effort and risk associated with continuous schema–performance evolution.

Limitations of the conceptual study. Because this work synthesizes existing literature rather than conducting new empirical experiments, its conclusions remain hypothetical and need empirical validation. The taxonomy and framework — while plausible and theoretically grounded — may not fully capture real-world complexity, especially in large, distributed, heterogeneous database ecosystems. Moreover, our prioritization heuristics and remediation strategy selection are abstract; concrete heuristics may differ across domains (e.g., OLTP vs data warehousing vs real-time analytics), workloads, data volumes, and organizational contexts. Finally, this study does not address human and organizational factors (e.g., team expertise, process discipline, cultural resistance) which often determine the success or failure of refactoring and performance initiatives.

Future research directions. We propose several avenues for further work:

- Empirical validation of the ISPE Framework via case studies in real-world systems: measure how combined schema refactoring + physical optimization affects maintainability, performance, and stability over time.
- Tool development: build or extend existing static analyzers, refactoring tools, and performance profilers into an integrated suite that implements the ISPE workflow.
- Integration into DevOps/CD pipelines: embed schema–performance audits and automated refactoring triggers into continuous integration/continuous deployment (CI/CD) pipelines, promoting agile schema evolution.
- Extension to multi-model and hybrid systems: apply or adapt the framework to systems combining relational, document, graph, or semi-structured data (e.g., JSON, XML), especially where flexible schemas co-exist with relational tables (Lahiri et al., 1999; Valduriez et al., 1986).
- Human and organizational study: investigate barriers and facilitators for adoption of continuous

schema–performance evolution, including developer practices, stakeholder incentives, and governance models.

CONCLUSION

In modern software ecosystems, database schemas and performance optimization often evolve along separate tracks, creating a disconnect that can inhibit scalability, maintainability, and long-term adaptability. This paper argues that schema quality and performance optimization should be integrated under a unified, continuous evolution framework. Through a structured synthesis of existing literature spanning schema design, anti-pattern detection, refactoring practices, and performance tuning — from traditional RDBMS like PostgreSQL to NewSQL and distributed systems — we propose the Integrated Schema–Performance Evolution (ISPE) Framework. This framework offers a taxonomy of schema and performance issues, maps them to remediation strategies, and prescribes a cyclical process of auditing, refactoring, optimization, testing, and monitoring. While conceptual, the ISPE Framework underscores the importance of bridging logical design and physical performance, and highlights trade-offs, challenges, and the promise of continuous, agile schema evolution. We advocate for future empirical validation, tool support, and operational integration to bring this vision into practice.

REFERENCES

1. Blaha, M. (2001). A retrospective on industrial database reverse engineering projects — part 2. In Proceedings Eighth Working Conference on Reverse Engineering, 147–153. IEEE.
2. Weber, J.H., Cleve, A., Meurice, L., & Ruiz, F.J.B. (2014). Managing technical debt in database schemas of critical software. In Sixth International Workshop on Managing Technical Debt, 43–46. IEEE.
3. Catalog of Database Refactorings (2019, December 21). Retrieved from http://www.agiledata.org/essays/databaseRefactoringCatalog.html
4. Eessaar, E. (2015). On query-based search of possible design flaws of SQL databases. In T. Sobh & K. Elleithy (Eds.), International Conference on Systems, Computing Sciences and Software Engineering (SCSS 12), 53–60. Springer, Cham.
5. Eessaar, E., & Voronova, J. (2015). Using SQL queries to evaluate the design of SQL databases. In T. Elleithy & T. Sobh (Eds.), International Conference on Systems, Computing Sciences and Software Engineering (SCSS 13), 179–186. Springer, Cham.
6. Khumnin, P., & Senivongse, T. (2017). SQL antipatterns detection and database refactoring process. In 18th IEEE/ACIS International Conference on Software Engineering, Artificial Intelligence, Networking and Parallel/Distributed Computing (SNPD), 199–205. IEEE.
7. Sharma, T., Fragkoulis, M., Rizou, S., Bruntink, M., & Spinellis, D. (2018). Smelly relations: measuring and understanding database schema quality. In Proceedings of the 40th International Conference on Software Engineering: Software Engineering in Practice, 55–64. ACM.
8. Delplanque, J., Etien, A., Auverlot, O., Mens, T., Anquetil, N., & Ducasse, S. (2017). CodeCritics applied to database schema: Challenges and first results. In 2017 IEEE 24th International Conference on Software Analysis, Evolution and Reengineering (SANER), 432–436. IEEE.
9. Vial, G. (2015). Database refactoring: Lessons from the trenches. IEEE Software, 32(6), 71–79.
10. SQL Performance Blog. (2021). Optimizing PostgreSQL Performance for High-Transaction Workloads. Retrieved from https://www.sqlperformance.com
11. Momjian, B. (2020). PostgreSQL: Introduction and Concepts. Addison-Wesley.
12. Cahill, M. J. (2009). Serializable Isolation for Snapshot Databases. University of Sydney.
13. Petkov, I. (2023). PostgreSQL Query Optimization Techniques. Apress.
14. Natti, M. (2023). 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.
15. Maia, F. C. M. B. O. (2015). Sharding by Hash Partitioning: A database scalability pattern to achieve evenly sharded database clusters. In 17th International Conference on Enterprise Information Systems (ICEIS), Barcelona, Spain.
16. Moniruzzaman, A. (2014). NewSQL: Towards Next-Generation Scalable RDBMS for Online Transaction Processing (OLTP) for Big Data Management. arXiv preprint.
17. Tankoano, J. (2023). Providing in RDBMSs the flexibility to work with various non-relational data models. Global Journal of Computer Science and Technology: H Information & Technology, 23(2).
18. Chen, P. (1976). The entity-relationship model — toward a unified view of data. ACM Transactions

on Database Systems, 1(1), 9–36.

19. Object Modeling Group. (2012). Unified Modeling Language Specification, Version 2.5.
20. Valduriez, P., Khoshajian, S., & Copeland, G. (1986). Implementation techniques of complex objects. In 12th International Conference on Very Large Data Bases (VLDB), Kyoto, August 1986.
21. Lahiri, T., Abiteboul, S., & Widom, J. (1999). Ozone: Integrating structured and semi-structured data. In 7th International Workshop on Database Programming Languages (DBPL): Research Issues in Structured and Semi-structured Database Programming, December 1999.