

International Conference on Internet of Things, Computing, and AI Applications
ICIoTCAA26-Conference Proceeding | IJMIET
Doi: [10.67228/30715628/ICIoTCAA-2026P105](https://doi.org/10.67228/30715628/ICIoTCAA-2026P105)
PP. 64-68

Original Article

SLM-Augmented PostgreSQL Query Optimization

Kiran Reddy Sudini

Director of Cloud Engineering & Operations, USA.

ABSTRACT

Database performance is one of the few problems that never stays solved. As data grows and access patterns shift, queries that were fast become slow, indexes that were right become wrong, and the tuning work begins again. Traditionally this work is manual, expert-intensive, and reactive. I designed an SLM-augmented query optimization pipeline for our PostgreSQL estate a small language model deployed inside our own network that continuously analyzes query workload and recommends rewrites, indexes, and planner hints. This article describes the design, and more importantly, the safety architecture that makes AI in the database layer responsible rather than reckless in a regulated healthcare environment.

KEYWORDS

Small Language Models, PostgreSQL, Query Optimization, Database Performance, SQL Query Rewriting, Index Recommendation, Query Planner Optimization, AI-Augmented Databases, Healthcare Database Systems, Safe AI Deployment.

1. THE PROBLEM: OPTIMIZATION DEBT AT SCALE

Our platform runs on PostgreSQL a zone-redundant primary with multiple in-region read replicas segregated by purpose (interactive API reads on some, reporting and batch on others) and a cross-region replica held for disaster recovery. Connection routing flows through server-side pooling and application-side pools. At this scale, with a workload spanning interactive traffic, reporting, and batch processing, query optimization is a continuous need rather than an occasional task. The telemetry to find problems exists statement statistics, execution plans from slow-query capture, and observability tooling but converting that telemetry into safe, validated improvements has historically required scarce senior database engineering time.

The insight behind the pipeline is that much of that conversion work follows recognizable patterns: an execution plan with a sequential scan over a large table and a selective predicate suggests a missing index; a query shape with a correlated subquery often rewrites to a join; a planner misestimate on a known distribution suggests a hint. These are exactly the pattern-recognition tasks that a code-capable language model can assist with if, and only if, the surrounding system guarantees that its suggestions can never do harm on their own.

2. WHY A SMALL LANGUAGE MODEL, AND WHY IN-NETWORK

The design deliberately uses a small language model a compact, code-oriented model in the several-billion-parameter class rather than a frontier-scale model behind a public API. The reasons are practical and regulatory. A small model is inexpensive enough to run continuously on our own Kubernetes infrastructure, entirely inside our virtual network. Nothing leaves the environment: no query text, no schema metadata, no telemetry is ever sent to an external AI service. In a HIPAA and HITRUST context, that single property in-network inference with no external dependency eliminates an entire category of data-handling risk and makes the compliance conversation tractable.

A smaller model also fits the task. Query rewriting, index recommendation, and hint generation are narrow, structured problems over a constrained vocabulary of SQL and plan features. They reward a model that is fast, cheap, and fine-tunable on our own workload far more than one that is maximally general.

3. THE CARDINAL RULE: OUT-OF-BAND, NEVER IN THE HOT PATH

The most important architectural decision is what the model is not allowed to touch. The SLM operates strictly out-of-band, as an advisor. It never sits in the query execution path, so no user request ever waits on model inference and a model failure can never degrade production traffic. And it never executes anything: the model does not hold credentials to run DDL, create indexes, or alter configuration. Its output is a recommendation with supporting evidence nothing more. This separation means the worst possible outcome of a wrong model output is a bad suggestion that a human declines, not a production incident.

The model advises; it never executes. That single constraint converts AI in the database layer from a hazard into a force multiplier.

4. THE PIPELINE END TO END

The pipeline runs as a sequence of stages, each with a defined responsibility:

- Telemetry collection. Statement statistics and captured execution plans are gathered continuously from the estate, identifying the queries where optimization effort will pay high total time, high frequency, or degrading trends.
- PHI-aware redaction. Before any query text reaches the advisor's datastore, literals are stripped and replaced with typed placeholders. The model reasons over query shape, never over data values. This is designed so protected health information cannot leak into the optimization system, its logs, or its training corpus.
- Inference. The model analyzes normalized query shapes together with plan features and schema metadata, and proposes candidate improvements: rewritten SQL, index recommendations, or planner hints.
- Automated validation. Every candidate is tested before a human ever sees it. Hypothetical-index tooling evaluates whether a proposed index would actually change the plan and estimated cost without building the index. Rewrites are checked for semantic equivalence and benchmarked against the original on a scrubbed replica, never against production.
- Human-gated review and apply. Validated recommendations, with their evidence plans before and after, cost deltas, benchmark results are queued for review by a database engineer. Only a human applies a change, through the normal change-management process, and every applied change carries its structured evidence for audit.
- Feedback and fine-tuning. Outcomes accepted, rejected, and post-apply performance flow back into a quarterly fine-tuning loop, so the model steadily specializes toward our schema, workload, and reviewers' standards.

5. VALIDATION IS THE HEART OF THE SYSTEM

It is tempting to treat the model as the interesting part of this design, but the validation layer is what makes the system trustworthy. Language models produce plausible output, and in database work plausible is not the standard correct is. Hypothetical-index evaluation answers, cheaply and objectively, whether the planner would even use a proposed index. Semantic-equivalence checking answers whether a rewrite returns the same results. Replica benchmarking answers whether the improvement is real under representative load. By the time a recommendation reaches a human, it is not a guess; it is a tested proposal with measured evidence. The human review then adds the judgment a model cannot: awareness of upcoming schema changes, write-amplification tradeoffs of new indexes, and operational context.

6. COMPLIANCE AS A DESIGN INPUT

Because the platform handles protected health information, the pipeline was designed compliance-first rather than retrofitted. The controls follow directly: literal redaction before storage so PHI never enters the advisor's world; in-network inference so nothing crosses the boundary; least-privilege service identities so the advisor can read telemetry but touch nothing else; human approval as a mandatory gate; and structured, per-change evidence that maps cleanly onto HITRUST audit

expectations. The result is a system an auditor can reason about: every optimization applied to production has a recorded origin, validation record, approver, and outcome.

7. ROLLOUT DISCIPLINE

The implementation is phased over roughly a quarter telemetry foundations first, then the redacting collector, then inference, then validation, then the review workflow, with the fine-tuning loop last so that each layer is proven before the next depends on it. Quantified results belong to real measurements rather than projections, so I will note simply that the targets are the classic ones: reduced mean query latency on the hot set, fewer emergency tuning escalations, and senior engineering time shifted from reactive firefighting to deliberate design

8. WHAT THIS REPRESENTS

I regard this pipeline as a template for how AI belongs in critical infrastructure: close to the data but outside the hot path, generative in its suggestions but powerless to act, validated by objective tooling, gated by human judgment, and auditable end to end. The database layer is among the least forgiving places to introduce AI which is precisely why doing it with this discipline matters. Applied this way, a small model becomes a tireless junior analyst for the database team: it reads every plan, notices every regression, drafts every candidate fix, and leaves the decisions to the people accountable for them.

REFERENCES

- [1] Kaoudi, Zoi, and Ioana Giurgiu. "Query Optimization Beyond Data Systems: The Case for Multi-Agent Systems." *arXiv preprint arXiv:2512.11001* (2025).
- [2] Yan, Zhengtong, et al. "DBMS-LLM Integration Strategies in Industrial and Business Applications: Current Status and Future Challenges." *arXiv preprint arXiv:2507.19254* (2025).
- [3] Qu, Ge, et al. "SHARE: An SLM-based hierarchical action CorREction assistant for text-to-SQL." *Proceedings of the 63rd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*. 2025.
- [4] Glenn, Parker, Alf Samuel, and Daben Liu. "Play by the Type Rules: Inferring Constraints for LLM Functions in Declarative ProgrSingh, Sarvjeet, et al. "Query Selectivity Estimation for Uncertain Database." (2007).
- [5] Singh, Sarvjeet, et al. "Query Selectivity Estimation for Uncertain Database." (2007).ams." *arXiv preprint arXiv:2509.20208* (2025).
- [6] Stonebraker, Michael, and Lawrence A. Rowe. "The design of Postgres." *ACM Sigmod Record* 15.2 (1986): 340-355.
- [7] Zhou, Feng, et al. "Nabil: A text-to-SQL model based on brain-inspired computing techniques and large language modeling." *Electronics* 14.19 (2025): 3910.
- [8] Sharma, Anantha, et al. "Mitigating hallucinationZeroG: An advanced knowledge management engine." *2025 International Conference on Artificial Intelligence, Computer, Data Sciences and Applications (ACDSA)*. IEEE, 2025.
- [9] Singh, Sarvjeet, et al. "Query Selectivity Estimation for Uncertain Database." (2007).
- [10] Platt, Nolan, and Pragyansmita Nayak. "Multi-Model Synthetic Training for Mission-Critical Small Language Models." *2025 3rd International Conference on Foundation and Large Language Models (FLLM)*. IEEE, 2025.
- [11] Sun, Wenbo, et al. "TransSQL+: Serving Large Language Models with SQL on Low-Resource Hardware." *Proceedings of the ACM on Management of Data* 3.6 (2025): 1-27.
- [12] Quoc Anh, Nguyen, et al. "Small Language Models For Efficient Domain Data Retrieval Using Structured Query Language from Natural Language." *Nguyen and Gia Hy, Truong and Ha, Son, Small Language Models For Efficient Domain Data Retrieval Using Structured Query Language from Natural Language (December 09, 2025)* (2025).
- [13] Shah, Sarvjeet Singh Chris Mayfield Rahul, and Sunil Prabhakar Susanne Hambrusch. "Query Selectivity Estimation for Uncertain Data." (2007).

- [14] Wang, Jing, et al. "Revisiting secondary indexing in {LSM-based} storage systems with persistent memory." *2023 USENIX Annual Technical Conference (USENIX ATC 23)*. 2023.
- [15] Caldeira, Carlos Eduardo, et al. "LLM-Based Text-to-SQL for SparkSQL Queries in Manufacturing Data Lakes: An Architecture for Industry 4.0." *2025 12th International Conference on Soft Computing & Machine Intelligence (ISCMI)*. IEEE, 2025.
- [16] Wang, Jing, et al. "Perseid: A secondary indexing mechanism for LSM-based storage systems." *ACM Transactions on Storage* 20.2 (2024): 1-28.