Um Framework Baseado em IA Generativa para Otimização de Queries em PostgreSQL: Um Estudo de Caso Industrial

Rafael F. Cardoso[1], **Leonardo C. Daronco**[2]

[1]Instituto de Informática – Universidade Federal do Rio Grande do Sul (UFRGS)
Caixa Postal 15.064 – 91.501-970 – Porto Alegre – RS – Brasil

[2]Mconf Tecnologia
Porto Alegre – RS – Brasil

rafael.cardoso@inf.ufrgs.br, daronco@mconf.com

Abstract. *The manual optimization of slow SQL queries in databases like PostgreSQL is a reactive and costly bottleneck for development teams. This paper proposes a framework that automates this cycle, using a Large Language Model (LLM) to analyze execution plans and generate actionable optimizations. An industrial case study validates the approach, demonstrating its effectiveness in proactively reducing query latency and optimizing engineering resources.*

Resumo. *A otimização manual de queries SQL lentas em bancos de dados como o PostgreSQL é um gargalo reativo e custoso para equipes de desenvolvimento. Este artigo propõe um framework que automatiza esse ciclo, utilizando um Modelo de Linguagem de Grande Escala (LLM) para analisar planos de execução e gerar otimizações acionáveis. Um estudo de caso industrial valida a abordagem, demonstrando sua eficácia na redução proativa da latência de queries e na otimização dos recursos de engenharia.*

1. Introdução

A performance de bancos de dados é um fator crítico para a responsividade de aplicações e a eficiência operacional [Yan et al. 2017]. Queries SQL ineficientes em produção causam degradação na experiência do usuário, aumentam custos de infraestrutura e podem levar à indisponibilidade de sistemas. A otimização tradicional dessas queries é um processo manual e reativo, dependente de especialistas com profundo conhecimento em otimizadores e na interpretação de planos de execução complexos, como os do comando `EXPLAIN ANALYZE` do PostgreSQL [The PostgreSQL Global Development Group 2025]. Essa expertise é escassa e cara, criando um gargalo operacional que desvia o foco dos desenvolvedores da construção de novas funcionalidades.

Para endereçar esse desafio, este artigo propõe um framework que automatiza a otimização de queries em PostgreSQL utilizando um Modelo de Linguagem de Grande Escala (LLM). A solução analisa proativamente as queries mais custosas, interpreta seus planos de execução e gera sugestões de otimização concretas e contextualizadas.

As principais contribuições deste trabalho são: (1) o projeto e implementação de uma arquitetura end-to-end para coleta e análise de queries lentas; (2) uma metodologia onde um agente de IA constrói contexto dinâmico (schema, índices, plano de execução) para garantir análises de alta fidelidade; e (3) um estudo de caso industrial que detalha a aplicação do framework e apresenta resultados quantitativos.

2. Fundamentação e Trabalhos Relacionados

Este trabalho se situa na interseção entre monitoramento de performance de bancos de dados e IA generativa para engenharia de software. Esta seção posiciona nossa solução frente ao estado da arte em ambas as áreas.

2.1. Monitoramento e Otimização de Banco de Dados

Soluções de mercado como *Datadog* e *pganalyze* oferecem observabilidade madura para bancos de dados, com análise de planos de execução e recomendação de índices [Datadog 2025, Duboce Labs, Inc. 2025]. Embora eficazes no diagnóstico, essas ferramentas apoiam um especialista humano em vez de automatizar a solução, e a implementação de melhorias permanece um processo manual. Nosso trabalho foca nesta lacuna entre diagnóstico e remediação, automatizando a transformação de dados de performance em código otimizado.

2.2. IA Generativa para Engenharia de Software

A aplicação de LLMs na engenharia de software foca predominantemente na tradução de linguagem natural para código [He et al. 2025], como no paradigma *Text-to-SQL* [Liu et al. 2025]. Nossa abordagem difere fundamentalmente ao partir de artefatos técnicos em vez de uma intenção humana. O LLM, portanto, atua não como tradutor, mas como um motor de raciocínio que analisa dados de diagnóstico para gerar transformações de código que otimizem a performance.

Este trabalho avança no uso de LLMs como motores de raciocínio, mas reconhece suas limitações, como a propensão a alucinações e a geração de código falho [Wang and Chen 2023, Liu et al. 2024]. Por isso, nossa arquitetura incorpora um mecanismo de validação com supervisão humana.

3. Arquitetura da Solução

A solução é um agente de IA baseado na plataforma n8n [n8n 2025], que orquestra a comunicação entre o banco de dados, uma API de LLM e um sistema de notificação. Sua arquitetura, ilustrada na Figura 1, é altamente configurável.

3.1. Coleta e Identificação de Queries

Periodicamente, o agente consulta a extensão `pg_stat_statements` para identificar as N queries mais custosas. A seleção combina o maior tempo total (`total_exec_time`) e médio (`mean_exec_time`) de execução, focando no impacto cumulativo do sistema e em gargalos que afetam a experiência do usuário.

3.2. Contextualização e Engenharia de Prompt

A eficácia do LLM depende do contexto que o agente constrói dinamicamente para cada query. Para uma análise precisa, o prompt inclui quatro artefatos do banco de dados: (1) a **query SQL original**; (2) o **schema das tabelas envolvidas**; (3) os **índices existentes** para evitar sugestões redundantes; e (4) o **plano de execução completo** (`EXPLAIN ANALYZE`). Essa contextualização permite ao LLM analisar a query em seu ambiente de execução real.

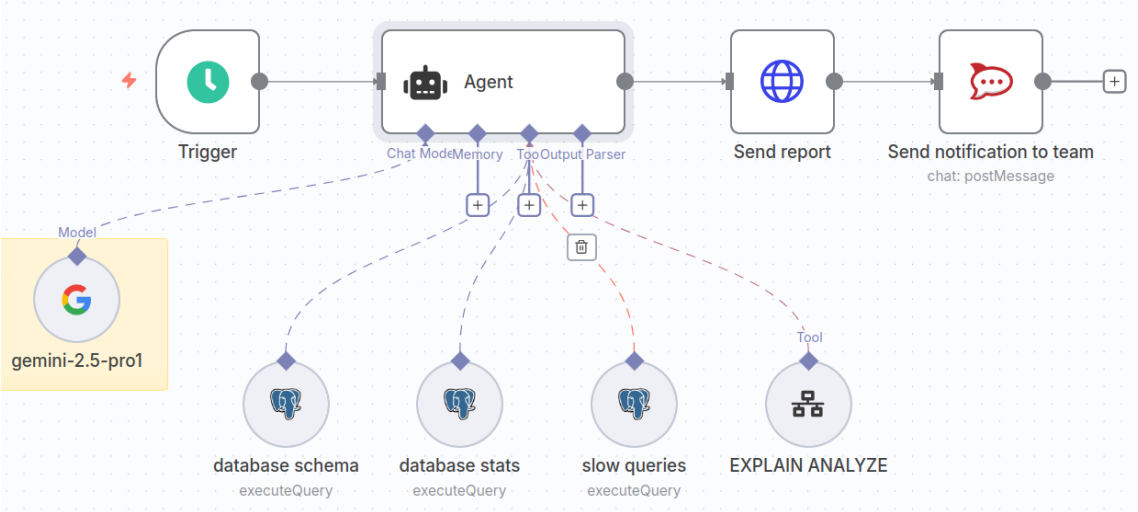

Figura 1. Arquitetura da solução

3.3. Geração da Otimização via LLM

O prompt contextualizado instrui o LLM (neste caso, Gemini 2.5 Pro) a atuar como um DBA especialista em PostgreSQL. O modelo deve: (1) analisar o plano de execução para identificar o gargalo; (2) explicar o problema concisamente; e (3) sugerir três soluções, como uma query SQL reescrita ou um comando DDL.

3.4. Framework de Validação com Supervisão Humana

Para mitigar riscos de alucinação, o framework adota o modelo *Human-in-the-Loop* (HITL), onde nenhuma alteração é executada automaticamente. As sugestões do LLM são formatadas em um relatório, enviadas a uma base de conhecimento, e a equipe responsável é notificada para revisão. Essa abordagem automatiza a demorada análise inicial, resolvendo um gargalo crítico e liberando os especialistas para a tarefa de maior valor: a validação e implementação segura da otimização.

4. Estudo de Caso: Aplicação e Resultados

O framework foi implantado por um mês na base de dados de produção da nossa aplicação, onde o agente identificou proativamente queries de alto custo. A Tabela 1 resume três otimizações representativas que foram validadas e implementadas.

Tabela 1. Resultados das otimizações sugeridas pelo agente

Problema	Latência	Sugestão Implementada	Latência	Redução
Uso de `IN` com subqueries aninhadas	332ms	Reescrita com `JOIN` explícito	95ms	-71%
Uso de `CTE` impedindo uso de índice	581ms	Substituição da `CTE` por `JOIN` com tabela derivada	190ms	-67%
Cláusula `OR` impedindo uso de índice	607ms	Substituição de `OR` por `UNION`	185ms	-70%

4.1. Análise dos Resultados

Os resultados demonstram o alto impacto do framework, com reduções de latência significativas, melhorando a performance dos endpoints, a experiência do usuário e a carga na infraestrutura. Qualitativamente, a equipe de desenvolvimento reportou que as sugestões foram consistentemente avaliadas como pertinentes e adotadas, confirmando o sucesso da abordagem.

5. Conclusão e Trabalhos Futuros

Este trabalho demonstrou a viabilidade e o impacto industrial de usar IA Generativa para automatizar a otimização de queries em PostgreSQL. O framework proposto transforma um processo manual e especializado em um sistema proativo, combinando dados de performance com o raciocínio de um LLM contextualizado. A principal lição é a importância crítica do contexto fornecido ao agente, destacando que a configuração de ferramentas é tão fundamental quanto a engenharia de prompts. Um desafio observado é que a complexidade do contexto exige LLMs de grande capacidade, pois modelos menores não processam satisfatoriamente o volume de informações.

Para trabalhos futuros, planejamos: (1) integrar o sistema com logs de aplicação para correlacionar queries lentas a chamadas de API; (2) adaptar a solução para outros SGBDs; (3) estender a análise à configuração do servidor PostgreSQL, como parâmetros de `autovacuum`; e (4) treinar LLMs especializados em otimização SQL para aumentar a precisão e reduzir custos.

Referências

Datadog (2025). Datadog. `https://docs.datadoghq.com`.

Duboce Labs, Inc. (2025). pganalyze. `https://pganalyze.com/docs`.

He, J., Treude, C., and Lo, D. (2025). Llm-based multi-agent systems for software engineering: Literature review, vision, and the road ahead. *ACM Trans. Softw. Eng. Methodol.*, 34(5).

Liu, F., Liu, Y., Shi, L., Huang, H., Wang, R., Yang, Z., Zhang, L., Li, Z., and Ma, Y. (2024). Exploring and evaluating hallucinations in llm-powered code generation.

Liu, X., Shen, S., Li, B., Ma, P., Jiang, R., Zhang, Y., Fan, J., Li, G., Tang, N., and 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(10):5735–5754.

n8n (2025). n8n: Secure workflow automation for technical teams. `https://n8n.io`.

The PostgreSQL Global Development Group (2025). Postgresql. `https://www.postgresql.org/docs`.

Wang, J. and Chen, Y. (2023). A review on code generation with llms: Application and evaluation. In *2023 IEEE International Conference on Medical Artificial Intelligence (MedAI)*, pages 284–289.

Yan, C., Cheung, A., Yang, J., and Lu, S. (2017). Understanding database performance inefficiencies in real-world web applications. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, page 1299–1308, New York, NY, USA. Association for Computing Machinery.