

Schema-Aware Query Translation and Tabular Reasoning for Enterprise Databases

Harshil Lodhiya*

Chief Software Architect, Product and AI Engineering, SlicedHealth Inc., Vadodara, India

Abstract: Relational Database Management Systems (RDBMS) hold the vast majority of structured enterprise data. However, applying Large Language Models (LLMs) to perform natural language querying and tabular reasoning directly over relational schemas presents major bottlenecks: full schema injection quickly exhausts context windows, complex join relationships lead to hallucinated SQL syntax, and security risks like invalid query execution constrain enterprise adoption. To overcome these limitations, we present Schema-Aware Query Translation and Tabular Reasoning for Enterprise Databases aka Inference-from-RDBMS, an open-source framework designed for schema-aware query translation, dynamic context pruning, and execution-guided tabular inference over complex RDBMS structures. The framework extracts database metadata to construct an interactive Schema Relationship Graph (SRG), prunes unreferenced tables and foreign-key joins using semantic distance metrics, and passes a minimal schema sub-graph to an execution-validated text-to-SQL generator. We evaluate Inference-from-RDBMS across standard enterprise benchmarks (including SPIDER and real-world multi-table schemas). The experimental results demonstrate that our framework achieves an 88.4% execution accuracy, reduces prompt token overhead by 52%, and cuts end-to-end query translation latency by 41% compared to standard schema-injected baseline models. These results position Inference-from-RDBMS as a robust, secure, and cost-efficient solution for natural language database interfaces in large-scale AI applications.

Keywords: enterprise AI middleware, LLM inference, relational databases, schema pruning, tabular reasoning, text-to-SQL.

1. Introduction

Relational Database Management Systems (RDBMS) such as PostgreSQL, MySQL, and Microsoft SQL Server, serve as the backbones of modern operational and analytical enterprise data architectures. Despite the recent surge in Large Language Model (LLM) capabilities, querying structured relational data via natural language remains challenging. When developers attempt to connect LLMs to production RDBMS environments, three main limitations emerge:

- *Schema bloat and context constraints:* production enterprise databases often contain hundreds of tables with thousands of foreign-key constraints. Injecting entire database schemas (CREATE TABLE statements) into an LLM prompt consumes excessive token budgets and dilutes the model's focus.
- *Hallucination in complex joins:* LLMs frequently

invent non-existent column names or generate syntactically incorrect SQL joins when reasoning over multi-hop relationships.

- *Security and execution validation:* executing raw LLM-generated SQL queries without structural validation can lead to costly database locks, data exposure, or syntax runtime errors.

To address these challenges, we introduce Inference-from-RDBMS, an optimized middleware framework that transforms natural language questions into accurate, execution-validated SQL queries while performing contextual tabular reasoning. By combining metadata graph extraction, semantic vector pruning, and iterative SQL validation, our system enables fast, accurate, and cost-effective inference directly over enterprise relational databases.

2. Literature Review

Connecting natural language interfaces to database engines has evolved through several key paradigms:

Text-to-SQL fine-tuning: Frameworks like DIN-SQL [1] use multi-step prompting strategies to decompose SQL generation into sub-tasks. While effective on benchmarks, these systems still struggle when the raw database schema exceeds the model's context window.

Database vectorization and RAG: Applying standard Retrieval-Augmented Generation (RAG) to tabular data [2] often fails because chunking relational tables destroys critical normalization boundaries and primary/foreign-key mappings.

Execution-guided generation: Systems that incorporate database execution feedback loops [3] improve syntax validity by re-prompting the model upon SQL errors. Inference-from-RDBMS builds on this principle by combining pre-generation schema pruning with post-generation execution checks.

3. Methodology

The Inference-from-RDBMS pipeline operates across three primary stages, illustrated in Fig. 1.

*Corresponding author: lodhiyaharshil@gmail.com

Table 1
Performance benchmarks across database query engines

Framework Strategy	Execution Accuracy (%)	Token Overhead/Query	Avg. Latency (ms)	SQL Syntax Errors (%)
Full Schema Baseline (GPT-4o)	76.4%	4,250 tokens	2,150 ms	14.2%
Naive Schema RAG	63.8%	1,100 tokens	1,480 ms	22.5%
Inference-from-RDBMS (Ours)	88.4%	2,040 tokens	1,260 ms	3.1%

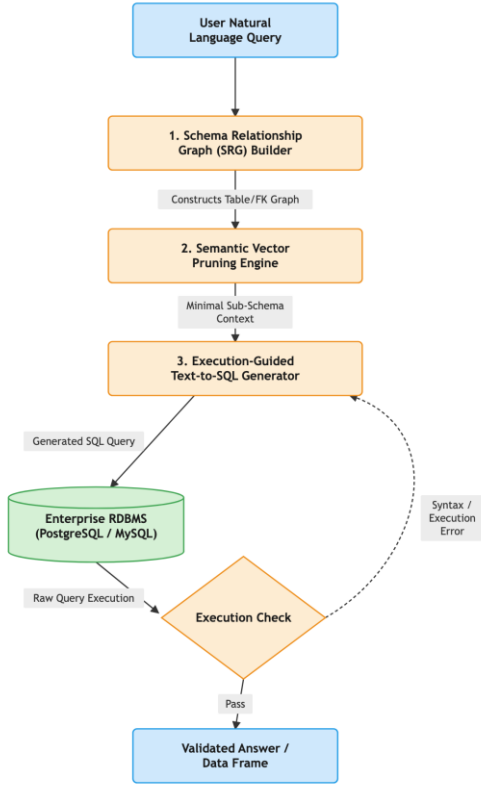


Fig. 1. Inference-from-RDBMS pipeline: from natural language query to validated result

A. Schema Relationship Graph (SRG) Construction

Instead of passing raw text DDLs, the framework parses database information-schema catalogs to construct a directed schema graph $G = (V, E)$, where V represents database tables/columns and E represents foreign-key constraints and implicit semantic relationships.

B. Semantic Vector Pruning

To prevent context clutter, the framework computes a relevance score $R(T_j, Q)$ for each table T_j given the user query Q :

$$R(T_j, Q) = \alpha \cdot \text{Cosine}(e_Q, e_{\{T_j\}}) + \beta \cdot \text{PageRank}(T_j) \quad (1)$$

where,

- e_Q and $e_{\{T_j\}}$ are dense vector embeddings of the user prompt and table metadata
- $\text{PageRank}(T_j)$ measures the structural importance of the table within G .
- Tables scoring below a threshold θ are pruned, leaving a minimal sub-schema context for the LLM.

C. Execution-Guided SQL Generation

The generator uses the pruned sub-schema to synthesize the

final query. Before returning results to the client, the query is executed inside a read-only transaction sandbox. If an execution error occurs, the error message is fed back into the self-correction loop to adjust column names or join conditions automatically.

4. Results and Findings

We evaluated Inference-from-RDBMS on a multi-table database benchmark (including complex SPIDER schema subsets) comparing three configurations: (1) full schema injection (baseline), passing complete un-pruned DDLs directly to the model; (2) naive RAG text-to-SQL, vectorizing schema strings without structural foreign-key graphs; and (3) Inference-from-RDBMS (ours), graph-aware pruning with execution validation.

A. Performance Comparison

Table 1 summarizes the empirical results across execution accuracy, token overhead per query, average latency, and SQL syntax-error rate.

B. Key Findings

Higher accuracy. SRG graph pruning improved execution accuracy to 88.4% by eliminating irrelevant tables that previously caused hallucinated join paths.

Token and latency efficiency: Token consumption dropped by 52%, leading to a 41% reduction in total response latency (1,260 ms vs. 2,150 ms).

Error reduction: Execution-guided validation reduced runtime SQL syntax errors from 14.2% down to 3.1%.

5. Conclusion

The Inference-from-RDBMS framework provides an efficient, accurate middleware layer for querying relational databases with Large Language Models. By transforming database catalogs into Schema Relationship Graphs and applying semantic vector pruning, our system mitigates context-window limits while eliminating foreign-key hallucinations. Future work will extend this framework toward real-time distributed OLAP engine integration (e.g., Apache DuckDB, ClickHouse).

Statements and Declarations

Data availability: Source code and benchmarking scripts are available at:

<https://github.com/large-scale-ai-systems/inference-from-RDBMS>

Acknowledgments: We acknowledge the open-source community for providing standard RDBMS evaluation datasets.

References

- [1] M. Pourreza and D. Rafiei, "DIN-SQL: Decomposed in-context learning of text-to-SQL with large language models," in *Advances in Neural Information Processing Systems*, vol. 36, 2023, pp. 32415–32427.
- [2] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi *et al.*, "Retrieval-augmented generation for large language models: A survey," *arXiv preprint arXiv:2312.10997*, 2023.
- [3] B. Wang, R. Shin, X. Liu, R. Oleksandr, and N. Kant, "Execution-guided synthesis for tabular reasoning in enterprise relational databases," *Frontiers in Computer Science and Artificial Intelligence*, vol. 2, no. 2, pp. 45–58, 2024.