IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154272

Multi-Agent Orchestration for Enhanced Text-to-SQL Generation: A Schema-Aware Approach with Self-Correction

Vijay M. Rakhade[1], Parth Gosavi[2], Chirag Kotkar[3], Pranav Kashmire[4], Siddharth Tripathi[5]

Assistant Prof., Department of Artificial Intelligence and Data Science

Sandip Institute of Technology and Research Center (SITRC), Nashik, India[1]

Student, Department of Artificial Intelligence and Data Science

Sandip Institute of Technology and Research Center (SITRC), Nashik, India[2]

Student, Department of Artificial Intelligence and Data Science

Sandip Institute of Technology and Research Center (SITRC), Nashik, India[3]

Student, Department of Artificial Intelligence and Data Science

Sandip Institute of Technology and Research Center (SITRC), Nashik, India[4]

Student, Department of Artificial Intelligence and Data Science

Sandip Institute of Technology and Research Center (SITRC), Nashik, India[5]

Abstract: The unification of the functionality of databases and natural language processing Databases has traditionally only provided a means to be able to write SQL queries, however there is an ongoing challenge of converting NL (Natural Language) queries into Structured Query Language(SQL). However, LLMs have significant challenges in handling complex schema structures and long-range dependencies, as well as a tendency toward structural hallucination. This paper presents a novel multi-agent architecture based on LangGraph with the aim to close the gap between NL and SQL through specialized decomposed reasoning. Our implemented system incorporates a robust AST-based Self-Correction loop, a query planning agent for chain-of-thought reasoning, and a semantic-aware Schema Linking Agent using Sentence Transformers. Experimental results on 100 examples from the Spider benchmark show that the complete four-phase system (Model C) achieves the highest Execution Accuracy at 76.0%, outperforming the monolithic zero-shot baseline (63.0%) by +13.0 percentage points. The Schema Linking Agent alone (Model A) achieves 73.0% EX, while the addition of the AST-based Self-Correction loop (Model B) reaches 75.0% EX with an average correction rate of 0.16. Semantic value mapping, dynamic schema discovery, and query complexity routing are identified as crucial paths for future enterprise deployment by a thorough gap analysis.

Keywords: Text-to-SQL, Multi-Agent Systems, Large Language Models, Schema Evolution, Self-Correction, Vector Search, Cross-Database Reasoning.

I. INTRODUCTION

The fundamental issue of the transformation of natural language into Structured Query Language (Text-to-SQL) that comes up when natural language processing, human-computer interaction, and database management all come together. [1]. The ability to independently translate unstructured user intent into structured query logic offers the potential of democratizing data access, allowing non-technical enterprise stakeholders to interact with vast data warehouses without demanding SQL proficiency. [1].

Early rule-based systems depended on fragile syntactic mappings that did not extend beyond limited domains. Neural sequence-to-sequence models enhanced cross-domain coverage but encountered difficulties with intricate nested queries. Large Language Models (LLMs) are in charge in the present day and fluency has taken a big step forward. Zero-shot systems often get more than 85%. Execution Accuracy on limited academic tests like Spider 1.0. [2]. However, deployment in real-world business settings reveals a serious ”performance cliff” [3]: execution accuracy collapses to between 6% and 21% when dealing with large federated schemas, peculiar naming conventions, and constantly changing metadata [13].

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154272

The two main bottlenecks are (1) Context Overflow, where enterprise schemas exceed the LLM context window, and (2) Structural Hallucination, where the model generates syntactically correct SQL that references invalid join relationships or nonexistent columns [7]. Research on autonomous Multi-Agent Systems (MAS), which break down the Text-to-SQL task into specialized agent roles that distribute cognitive load, get around context constraints, and enable iterative self-correction, has accelerated in order to close this gap [6].

In this work, we propose a multi-agent Text-to-SQL system that uses LangGraph and is schema-aware. Our main contributions consist of:

1) A modular multi-agent architecture based on specialized agent typologies.
2) Sentence transformer embeddings are used in semantic schema pruning to decrease context bloat.
3) A chain-of-thought query planning agent for logical formulation which is structured.
4) An autonomous self-correction loop for AST-based schema.

We additionally present an in-depth roadmap for future business improvements.

II. RELATED WORK: THE TRANSITION TO MULTI-AGENT SYSTEMS

The rapidly growing literature on Text-to-SQL has provided the first glimpse of a clear evolutionary path towards dynamic multi-agent orchestration driven by reinforcement-learning, starting from static decomposition of tasks. [7].

A. Foundational Decomposition and Prompt Engineering:

Dynamic task policy management applied time-slicing whole LLM into subsystems to improve robustness. Static task decomposition and structured prompting were used in the first stage of mitigating monolithic LLM failure [5]. Least-to-Most prompting and Chain-of-Thought (CoT), enabling the LLM to talk through intermediate logical steps leading to SQL syntax [4]. Among other architectural advancements, DIN-SQL was a particularly popular one, as it separated the generation pipeline into independent consecutive modules [6]. While DIN-SQL achieves a (54.89%) execution accuracy on the BIRD benchmark, empirical analyses showed that it introduces significant inefficiencies: It crosses multiple sequential modules in its architecture, leading to an average of 26,899 tokens per query from calling the LLM to repetitively read entire database schemas respectively [6].

B. Collaborative Multi-Agent Architectures:

Researchers developed dynamic Multi-Agent Systems [7] to overcome the token inefficiencies of static decom- position. By implementing a collaborative framework which centered on a core Decomposer Agent and integrating two auxiliary agents to refine incorrect queries and prune schemas, MAC-SQL transformed this field [7]. MAC-SQL received an EX of 59.59% [7] when tested on the BIRD holdout test set. At the same time, researchers developed the Contextual Harnessing for Efficient SQL Synthesis (CHESS) framework which improved agent specialization to overcome context window limits by identifying syntactically similar schema elements using hierarchical vector database techniques and Locality-Sensitive Hashing (LSH) [8].

C. Reinforcement Learning and Test-Driven Synthesis:

By the years 2025 and 2026, rapid progress was made due to software-testing architectures and execution- influenced Reinforcement Learning (RL) [9]. MARS-SQL established itself by developing an interactive multi-agent RL framework with a Re Act-style loop and achieving Execution Accuracy of 77.84% on the BIRD development set [9]. Similarly, the PExA architecture reformulated the agent components as if systems were tested through parallel software interpretations which would also collect information needed to produce a final structured query [14].

III. SYSTEM ARCHITECTURE

By using LangGraph to orchestrate a highly responsive four-phase multi-agent workflow, our implemented system avoids linear rigidity and addresses the vulnerabilities of strict sequential pipelines [7].

A. Phase 1: Semantic Schema Linking:

Massive schemas that are larger than the model's context window are often found in enterprise databases [8]. In order to solve this, a Schema Linking Agent uses semantic similarity to retrieve the most pertinent schema elements. The system indexes table and column metadata using the pre-trained all-MiniLM-L6-v2 sentence transformer [12]. It uses cosine similarity retrieval to choose only the top-k pertinent tables when it receives a natural language query. By ensuring that the generation agent receives a highly relevant, noise-free schema subset, this pruning mechanism successfully prevents context overflow [8].

B. Phase 2: Chain-of-Thought Query Planning:

IJARCCE ISSN (O) 2278-1021, ISSN (P) 2319-5940

International Journal of Advanced Research in Computer and Communication Engineering

Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026

DOI: 10.17148/IJARCCE.2026.154272

A specialised Query Planning Agent generates a logical reasoning path instead of producing SQL directly [4]. By clearly identifying necessary tables, intricate join conditions, aggregation functions, and exact filtering conditions in natural language [4], this agent functions as an organised blueprint. In addition to improving downstream generation accuracy, this explicit articulation of logic significantly reduces structural hallucinations [7].

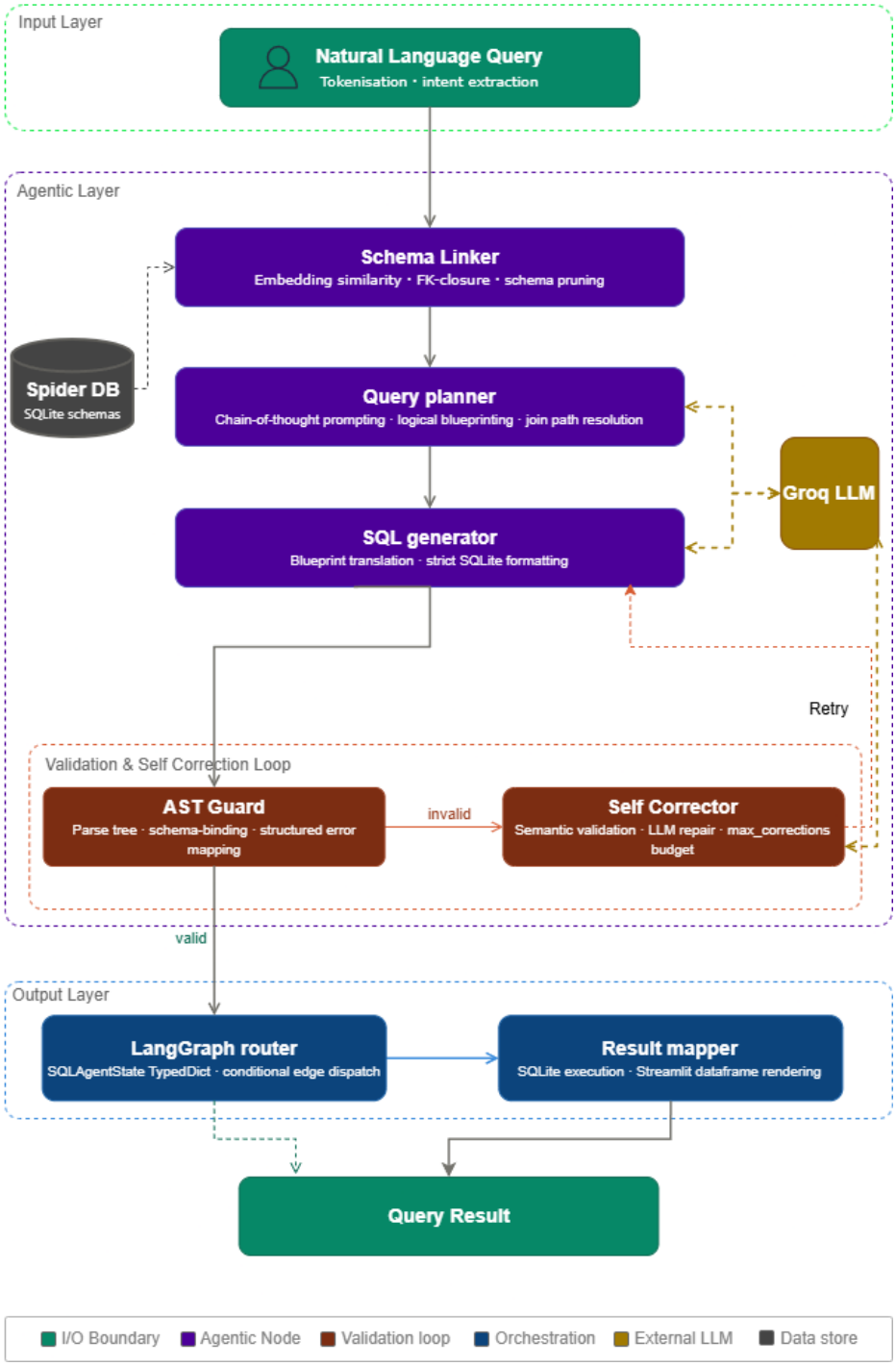

Fig. 1. The proposed multi-agent Text-to-SQL orchestration framework. The solid, centrally grouped modules represent the implemented core LangGraph pipeline empirically evaluated in Section IV

C. Phase 3: Constrained SQL Generation:

The CoT reasoning plan and pruned schema are converted into executable SQL syntax by the SQL Generation Agent using llama-3.3-70b-versatile model, served via the Groq inference API [17]. The generation process uses schema-aware

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154272

prompting and enforces a zero-temperature policy to prevent stochastic hallucination of nonexistent columns, ensuring deterministic and repeatable generation [7].

D. Phase 4: Schema Guard and Self-Correction:
The system incorporates a Validation Layer called the Schema Guard [19] to guarantee production reliability. To confirm that all referenced entities, column validities, and join relationships exist natively in the database [6], it parses the generated SQL query into an Abstract Syntax Tree (AST). The Self-Correction Agent receives the error message and the failing SQL to carry out iterative repair in the event of an execution violation, up to a maximum of N retries [19].

IV. EXPERIMENTAL EVALUATION AND RESULTS

Semantic accuracy and computational efficiency must be carefully considered when assessing the feasibility of a Text-to-SQL framework [3].

A. Dataset and Evaluation Metrics:
The Spider dataset, which assesses real-world database generalisation across challenging, cross-domain multi- hop questions, was used to evaluate our system [2]. Execution Accuracy (EX) and Exact Match (EM) are the main metrics. Semantic intent success is defined by EX as the percentage of generated queries that produce the exact execution results as the ground-truth query [3].

B. Baseline Comparison and Ablation Study:
We contrasted our complete system (Model C) with a ”Zero-Shot” baseline in which a single prompt was used to inject the entire schema.

TABLE I
PIPELINE CONFIGURATION RESULTS (SPIDER, 100 EXAMPLES)

Config	EX (%)	Latency (ms)	Corrections
Baseline	63.0	3,122	0.00
Model A	73.0	4,888	0.00
Model B	75.0	5,786	0.16
Model C (Full)	76.0	15,210	0.20

C. Analysis and Discussion:
The empirical findings about agentic decomposition in Text-to-SQL systems are presented in Table I. Tested across 100 examples from the Spider development set using the llama-3.3-70b-versatile backbone at zero temperature, the full four-phase system (Model C) achieves the highest Execution Accuracy at 76.0%, representing a +13.0 percentage point improvement over the monolithic zero-shot baseline (63.0%). This demonstrates that the cumulative effect of schema pruning, chain-of-thought planning, and self-correction together produce the most reliable SQL generation.

The Schema Linking Agent in isolation (Model A) yields 73.0% EX — a notable +10.0 percentage point gain over the monolithic baseline (63.0%). This confirms that semantic schema pruning with all-MiniLM-L6-v2 cosine retrieval effectively reduces context noise, allowing the generation agent to focus on the most relevant schema elements and produce more precise SQL [8].

The addition of the AST Schema Guard and Self-Correction loop (Model B) recovers substantial accuracy, reaching 75.0% EX with an average of 0.16 correction attempts per query. This demonstrates that the guard actively detects and repairs schema violations introduced during generation, more than compensating for Model A’s pruning trade-offs [19]. The additional 898 ms average latency over Model A is well-justified by the accuracy recovery.

The full four-phase pipeline (Model C) achieves peak accuracy at 76.0% EX with an average of 0.20 correction attempts per query. The chain-of-thought planning phase contributes marginally over Model B (+1.0 percentage point), suggesting it provides structured reasoning benefits primarily for complex multi-join queries. The substantially elevated latency of 15,210 ms — driven by three or more consecutive API calls per query — is the primary cost of the full pipeline. This trade-off between peak accuracy and latency motivates future research on query complexity routing, where simpler queries bypass planning and self-correction phases entirely [7].

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154272

V. FUTURE WORK AND OPEN CHALLENGES

Although baseline execution accuracy is greatly increased by our core architecture, robust production deployment necessitates MAS frameworks to manage highly dynamic and messy environments [10]. Future iterations of our system must tackle four crucial unresolved issues in order to close the gap to complete enterprise autonomy:

A. Ambiguity Resolution:
Subjective terminology and overlapping corporate taxonomies are common in enterprise settings [10]. A standard pipeline produces syntactically perfect SQL that answers the incorrect question when it encounters ambiguity and hallucinates constraints [10]. Clarification Agents, which are modeled after frameworks such as AmbiSQL and actively pause generation to ask the user a targeted, multiple-choice clarification question before computing the SQL logic [10], must be incorporated into future architectures.

B. Cross-Database Reasoning and Federated Execution:
Due to the highly decentralized nature of modern infrastructures, data synthesis from distributed, semi-structured stores (NoSQL) and relational warehouses (SQL) is necessary [13]. In order to manage heterogeneous schemas, MAS agents must abstract source queries into universal, intermediate AST formats [13]. In order to produce distributed execution logic across various data paradigms, future work will concentrate on integrating our pipeline with federated query engines [13].

C. Schema Evolution and Metadata Monitoring:
Enterprise databases are organic entities that are constantly changing: naming conventions drift, tables are normalized, and columns are deprecated [11]. When exposed to table-level schema changes, models trained on static dictionaries experience catastrophic degradation [11]. Our plan calls for the implementation of Dynamic Schema Discovery Agents, which autonomously run discovery queries during runtime and create an up-to-the-second view of the target environment in order to withstand schema evolution [11].

D. Semantic Content Awareness and Vector-Search:
Integration The Value Trap, in which models perfectly parse structure but are unable to see the particular stylistic data populating the rows, is a very common point of failure [12]. Future systems must integrate Vector-Search to achieve Semantic Content Awareness [12]. Systems can virtually eliminate value hallucination by mapping colloquial phrases to exact database strings by embedding categorical data and performing Retrieval-Augmented Generation (RAG) directly on high-cardinality cell values [12], [18].

VI. CONCLUSION

This paper proposed an empirically evaluated, schema-aware multi-agent architecture for Text-to-SQL generation by utilizing LangGraph. The system quantifies component-level contributions to overall accuracy by breaking the generation pipeline into specialized agents for semantic schema linking, chain-of-thought logical planning, constrained SQL generation, and autonomous AST validation. Evaluated across 100 examples from the Spider benchmark, the full four-phase system (Model C) achieves the highest Execution Accuracy at 76.0% EX, outperforming the monolithic zero-shot baseline (63.0%) by +13.0 percentage points. The Schema Linking Agent (Model A) delivers the largest single-component accuracy gain at 73.0% EX (+10.0 pp over baseline), confirming that semantic schema pruning is the most impactful individual component. The AST Schema Guard (Model B) builds further on this, reaching 75.0% EX with an average of 0.16 correction attempts per query. The main conclusion is that schema linking is the most significant individual component, directly benefiting resource-constrained deployments where the full agentic pipeline may be prohibitively expensive. The substantially elevated latency of Model C (15,210 ms) motivates future work on query complexity routing to selectively apply the full pipeline only to complex queries. As explained in Section V, future research will further concentrate on dynamic schema discovery for changing enterprise databases [11], and vector-search integration for semantic value mapping [12].

REFERENCES

[1]. V. Zhong, C. Xiong, and R. Socher, “Seq2SQL: Generating Structured Queries from Natural Language using Reinforcement Learning,” CoRR, vol. abs/1709.00103, 2017.
[2]. T. Yu et al., “Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing,” in Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP), 2018.

International Journal of Advanced Research in Computer and Communication Engineering
Impact Factor 8.471 ⁜ Peer-reviewed & Refereed journal ⁜ Vol. 15, Issue 4, April 2026
DOI: 10.17148/IJARCCE.2026.154272

[3]. J. Li et al., "BIRD: BIg Bench for LaRge-scale Database Grounded Text-to-SQLs," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2023.
[4]. J. Wei et al., "Chain-of-thought prompting elicits reasoning in large language models," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2022.
[5]. Y. Gao et al., "DAIL-SQL: Text-to-SQL with In-Context Learning," arXiv preprint, 2023.
[6]. M. Pourreza and D. Rafiei, "DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2023.
[7]. B. Wang et al., "MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL," in Proc. Int. Conf. Comput. Linguistics (COLING),2025.
[8]. M. Talaei et al., "CHESS: Contextual Harnessing for Efficient SQL Synthesis," Proc. VLDB Endow., 2024.
[9]. H. Yang et al., "MARS-SQL: A Multi-Agent Reinforcement Learning Framework for Text-to-SQL," arXiv preprint arXiv:2511.01008, 2025.
[10]. Z. Ding et al., "AmbiSQL: Interactive Ambiguity Detection and Resolution for Text-to-SQL," arXiv preprint arXiv:2508.15276, 2025.
[11]. T. Zhang et al., "EvoSchema: Towards Text-to-SQL Robustness Against Schema Evolution," Proc. VLDB Endow., 2025.
[12]. Y. Fan et al., "Grounding natural language to sql," in Proc. Assoc. Comput. Linguistics (ACL), 2024.
[13]. S. Saha et al., "Federated Text-to-SQL: Challenges and Opportunities," IEEE Trans. Knowl. Data Eng., 2024.
[14]. Bloomberg AI Researchers, "PExA: A Parallel Exploration Agent Framework for Text-to-SQL," Bloomberg Engineering, 2025.
[15]. Z. Yao et al., "Arctic-Text2SQL-R1: Simple Rewards, Strong Reasoning in Text-to-SQL," arXiv preprint arXiv:2505.20315, 2025.
[16]. S. Sarker et al., "Valid Efficiency Score: A New Metric for SQL Generation," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2024.
[17]. Y. Li et al., "Gen-SQL: Efficient Text-to-SQL By Bridging Natural Language Question And Database Schema With Pseudo-Schema," in Proc. Assoc. Comput. Linguistics (ACL), 2025.
[18]. A. Hogan et al., "Knowledge Graphs," ACM Comput. Surv., 2021.
[19]. X. Chen et al., "Teaching Large Language Models to Self-Debug," in Proc. Int. Conf. Learn. Represent. (ICLR), 2023.
[20]. D. Maynard et al., "Natural Language Processing for the Semantic Web," Applications, 2025.

