Chatting With Your Data: LLM-Enabled Data Transformation for Enterprise Text to SQL

VULQAN.ai
Correspondence:
vikas.kapoor@vulqan.ai

Sara Mani Kapoor
University of Cambridge
Department of Computer Science and Technology
smk78@cam.ac.uk

Dr. Soumya Banerjee
University of Cambridge
Department of Computer Science and Technology
sb2333@cam.ac.uk

Abstract

Recent advances in large language models (LLMs) have shown remarkable success in code generation across general-purpose programming languages. However, the translation of natural language to SQL in real-world enterprise settings remains a challenge. While state-of-the-art models achieve over 80% execution accuracy on academic datasets like Spider and BIRD, their performance drops dramatically in the face of large, heterogeneous enterprise schemas. This gap stems not from SQL's syntactic complexity but from the implicit business knowledge and schema-level irregularity embedded in real enterprise databases. In this work, we present **MAIA**, a **M**anagement **A**bstraction and **I**ntelligence **A**lgorithm that transforms fragmented enterprise data models into semantically enriched logical representations. These logical data models (LDMs) represented in a knowledge graph guide LLM reasoning through schema abstraction, synonym resolution, and business logic alignment. Our method reframes Text-to-SQL as a knowledge representation problem and introduces a sequential agent-based framework that orchestrates aspects like object and variable selection, condition inference, and query assembly. We evaluate our approach on several real-world benchmarks reflecting salient use cases across industries. Our framework significantly outperforms standard prompting baselines on models like LLaMA-3-70B-Instruct and GPT-o3, especially on queries involving joins, nested logic, and ambiguous semantics. This work highlights the importance of schema intelligence and suggests that the most impactful innovations in industry Text-to-SQL systems may lie not in code synthesis, but in making the underlying structured data representation more logical and explainable.

1 Introduction: Motivation & Contributions

Large Language Models (LLMs) have achieved remarkable success in code generation, with models like Codex, AlphaCode, and PaLM-Coder reporting over 70% success rates in translating natural language to functional code [Ghahramani, 2023] [Chen et al., 2021] [Li et al., 2022]. SQL presents a particularly compelling target for LLM augmentation since it's used by 51% of professional developers yet only 35.3% have received formal training [Stack Overflow Developer Survey Team, 2024] [Stack Overflow, 2022]. Indeed, over the past few decades the field of "Text-to-SQL" has surged with dozens of publications detailing a variety of LLM-driven methodologies. Much of this research paints an optimistic picture with SOTA systems surpassing 80% execution accuracy on common benchmarks.

Despite these promising statistics, recent research reveals a stark reality gap. In late 2024, Spider 2.0 showed that the very SOTA models scoring 80%+ on academic benchmarks plunge below 10% when

39th Conference on Neural Information Processing Systems (NeurIPS 2025) Workshop: Generative AI in Finance.

evaluated on real enterprise data Lei et al. [2025]. This gap indicates that success on well-structured, academic datasets does not transfer to frenetic enterprise schemas, where the core challenge lies less in SQL syntax and more in schema-level irregularity: hundreds of tables & columns, inconsistent naming, sparse documentation, missing foreign keys, unnormalized lists, implicit joins that demand domain knowledge, and ambiguous natural-language (NL) queries. Benchmarks must explicitly model these nuances for Text-to-SQL to be useful in real deployments.

This paper addresses the enterprise gap by reframing Text-to-SQL as a knowledge representation problem rather than pure syntax parsing, with the central insight that LLM failures stem from lacking access to the implicit business knowledge buried in enterprise schemas. We propose **MAIA**, a **M**anagement **A**bstraction and **I**ntelligence **A**lgorithm that converts fragmented schemas into logically consistent, semantically rich representations formalized as a graph. This abstraction layer is, we argue, essential to LLM-based Text-to-SQL reasoning. More formally, in this work we contribute a **real-world benchmark** paired with $\sim$60 NL questions that reflect ambiguities expected in ordinary business queries. We provide a **schema-to-knowledge graph pipeline** that structures the source schema into a unified semantic reasoning layer via entity abstraction, synonym resolution, and relationship inference. Further, we introduce an **agent-based query generation framework** that orchestrates modular agents for intent parsing, schema linking, and SQL assembly to improve compositional accuracy over vanilla prompting. Injecting this contextualized schema view into prompts yields significant accuracy gains on real queries.

2 Related Work and Background

2.1 Non-Enterprise Approaches & Benchmarks

Modern LLM-enabled Text-to-SQL methods can be broadly categorized into four primary areas: prompt-based in-context learning (e.g. DIN-SQL), fine-tuning (e.g. CodeS), agentic refinement (e.g. CHESS), and metadata-enriched modeling (e.g. Knowledge-to-SQL) Pourreza and Rafiei [2023], Li et al. [2024], Talaei et al. [2024], Hong et al. [2024]. Though these approaches typically quote accuracies of over 60%, most are evaluated on small, well-organized datasets with little real-world complexity. Spider, for instance, a well-known benchmark leader averages 5.1 tables and 27.6 columns, with manual FKs and transparent names. Other commonly used datasets like BIRD and WikiSQL are similarly designed to minimize imperfections, yielding an overly optimistic picture of real-world performance Yu et al. [2018], Zhong et al. [2017], Li et al. [2023a].

```
Question 18:  What is the average and maximum capacities for all stadiums ?
SQL:  select avg(capacity) ,  max(capacity) from stadium
```

Figure 1: An example of a natural language–SQL pair from the Spider dataset

Questions in existing benchmarks are carefully constructed to eliminate ambiguity. Spider for instance explicitly avoids questions that are “vague or too ambiguous" Yu et al. [2018]. Questions like that in Figure 1 create artificial lexical overlap directly mapping to SQL syntax.

2.2 Enterprise-Oriented Approaches & Benchmarks

Enterprise-focused Text-to-SQL research remains extremely limited, with the groundbreaking Spider 2.0 benchmark published only in November 2024 Lei et al. [2025]. Notable approaches like ReFoRCE20’s retrieval-augmented reasoning system Deng et al. [2025] and Sequeda et al.’s knowledge graph-augmented methods that achieve significant accuracy improvements on enterprise questions Sequeda et al. [2023] validate that semantic enrichment is essential for enterprise success. However, even these advances have limitations, as many enterprise benchmarks on which these methods are tested still prioritize evaluation consistency over the messiness characteristic of real-world deployments. Though Spider 2.0’s benchmark advances enterprise realism with 632 workflow tasks from various companies, Lei et al. [2025] its “lite” variants like Spider-snow still employ semantically transparent naming conventions such as the `IMDB_MOVIES` database with intuitively named tables like `MOVIES` and columns like `avg_rating`.

Like non-enterprise benchmarks, the NL questions in the Spider 2.0 dataset maintain close alignment with SQL. Although their goal of “removing ambiguity in the expected results and ensuring that all

SQL conditions are clearly mentioned" enables consistent evaluation, it creates an artificial clarity that may not reflect enterprise reality Lei et al. [2025].

Despite methodological progress and sophisticated benchmarks, current Text-to-SQL research (both academic and enterprise) inadequately prepares systems for real-world environments where production databases are replete with ambiguity and inconsistency.

3 Dataset

The benchmark we present simulates a real analytic use-case that requires reasoning across disparate data sources. The use-case methodology directly emulates how business analysts would need to "chat with data" that likely spans multiple sources. This Ticket Management use-case is derived from a technology firm managing software development tickets across three system transitions: Excel, Jira, and a homegrown workflow application. The dataset contains three tables with no foreign keys, inconsistent priority encodings across systems, and ambiguous field interpretations. Accompanied by approximately 60 NL questions authored with non-obvious schema alignment, navigating the dataset entails question disambiguation, relationship inference, and list normalization across sources. Questions are classified by schema reasoning complexity: Level 0 (single-table lookups), Level 1 (single-table with filtering/grouping/list normalization), Level 2 (multi-table joins), and Level 3 (exploratory questions with multiple plausible interpretations). To protect proprietary data while preserving schema complexity, all identifiable information was anonymized with manually renamed tables and columns, and synthetic data values generated with an LLM using original distributions and formats.

4 Methodology

We introduce **MAIA** (**M**anagement **A**bstraction and **I**ntelligence **A**lgorithm), an LLM-enabled data transformation methodology that converts messy multi-source data into semantically aligned representations and substantially improves SQL generation performance in enterprise environments. Our system consists of a multi-agent data orchestration framework that turns an incoming Source Data Model (SDM) which is the original source schema into a Logical Data Model (LDM) which is an abstracted representation of the SDM in a semantic Knowledge Graph (KG).

The LDM is derived via a transformation layer built on VULQAN.ai's proprietary Canonical Data Model (CDM), a curated ontology of core business entities and variables across industries. This CDM acts as a semantic template for methodically creating LDMs out of SDMs. In the following, we show for the Ticket Management use case how the LDM is created and utilized to enhance SQL generation.

4.1 From Source Data Model to Logical Data Model

In Ticket Management, the LDM aligns objects across Jira, Excel, and a homegrown system and maps variables from each source to logical entities: **Core entities**: `Employee`, `Ticket`, `Environment`, `Project`, `Release`; **Keys:** Composite keys per object plus a proprietary primary key; synthetic foreign keys generated where missing; **Naming normalization:** e.g., `SRC` $\rightarrow$ `Source`, `Rls` $\rightarrow$ `Release`; align semantically equivalent fields across sources (e.g., `Status` = `Dev Status`, `Date Opened` = `Created`); **List harmonization:** Ticket statuses from multiple sources (e.g., Jira: `Open/Reopened, In Progress, Done, Closed`; Excel: `Initiated, Submitted, Completed, Closed, Merged, Withdrawn`; etc.) are normalized to a canonical list; **Relationship roles:** Infer from variable names (e.g., `Employee` can have the role of a `Project Lead` in the relationship to the `Ticket` table); **Reshuffle:** Move relevant variables into their objects with required PK/FK scaffolding.

4.2 Schema-Enriched Knowledge Graph Representation

The LDM is represented Neo4j graph instance where core entities are nodes, relationships are directed with domain roles. Node properties include relevant attributes and metadata. The graph provides LLM agents a structured traversal space for accurate schema linking.

4.3 Agent Architecture, Orchestration, and Reasoning

MAIA agents first clarify intent and ground the query in the LDM. Then they proceed by mapping phrases to objects in the KG, then selecting variables, deriving appropriate filters and ultimately

assembling the SQL using joins implied by the LDM. The SQL is then repaired using DB messages to produce a final dialect-correct, schema-aligned query.

5 Experiments

Experiment1 benchmarks GPTo3-mini, LLaMA3-70B-Instruct, and Phi-4-14B zero-shot; Experiment 2 repeats with only open models for governance. Baselines use minimal prompts over the SDM, and evaluation adopts an LLM-as-a-Judge semantic rubric with deterministic settings and normalization to mitigate execution-accuracy pitfalls (spurious matches, underspecified queries), grading outputs as Pass/OK/Fail.

6 Results & Discussion

As expected, baseline results using vanilla prompts were poor across all tested models, averaging barely above 10% accuracy on the Ticket Management use case with performance dropping to as low as 0% for Level 2-3 questions requiring joins or exploratory analysis. This performance on real-world data is notably worse than Spider 2.0's baseline of 23.4% and far below BIRD's 54% accuracy rates Lei et al. [2025] Li et al. [2023b]. Our results likely overestimate true enterprise performance since production databases involve datasets orders of magnitude larger and more complex than our Ticket Management case study.

LLM	Use Case	Level 0	Level 1	Level 2	Level 3	Overall
Phi-4-14B	Ticket Management	83%	38%	0%	0%	14%
LLaMA-3-70B	Ticket Management	83%	0%	7%	0%	12%
GPTo3-mini	Ticket Management	100%	13%	7%	0%	15%

Table 1: LLM Accuracy by Difficulty Level (Baseline)

LLM	Use Case	Level 0	Level 1	Level 2	Level 3	Overall
Phi-4-14B	Ticket Management	86%	71%	75%	53%	69%
LLaMA-3-70B	Ticket Management	100%	100%	86%	47%	78%

Table 2: LLM Accuracy by Difficulty Level (MAIA)

Our experimental evaluation demonstrates that the **MAIA** framework substantially outperforms Text-to-SQL baselines, with LLaMA-3-70B achieving 78% accuracy on the Ticket Management use case compared to baseline performance. Qualitative analysis reveals that **MAIA** produces sophisticated queries with advanced SQL constructions including CTEs, complex CASE statements, and intelligent normalization, demonstrating methodical navigation through semantically opaque schemas using our abstraction layer.

7 Conclusion and Future Work

This study demonstrates the significant potential for LLM-enabled data transformation in addressing complex enterprise Text-to-SQL challenges through **MAIA**'s semantic abstraction layer. **MAIA** provides not only functional accuracy but also interpretable, explainable query construction that both technical data scientists and non-technical business analysts can understand and validate. This transparency represents a crucial advantage for enterprise deployment, where analysis must often be audited, modified, or explained to stakeholders with varying technical backgrounds. Ultimately through this research we establish foundational components for more intelligent data orchestration, Text-to-SQL translation and user dialog with enterprise data across industries and use-cases.

References

Zoubin Ghahramani. Introducing palm 2. https://blog.google/technology/ai/google-palm-2-ai-large-language-model/, June 2023. Google AI Blog; updated September 12, 2023.

Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, Alex Ray, Raul Puri, Gretchen Krueger, Michael Petrov, Heidy Khlaaf, Girish Sastry, Pamela Mishkin, Brooke Chan, Scott Gray, Nick Ryder, Mikhail Pavlov, Alethea Power, Lukasz Kaiser, Mohammad Bavarian, Clemens Winter, Philippe Tillet, Felipe Petroski Such, Dave Cummings, Matthias Plappert, Fotios Chantzis, Elizabeth Barnes, Ariel Herbert-Voss, William Hebgen Guss, Alex Nichol, Alex Paino, Nikolas Tezak, Jie Tang, Igor Babuschkin, Suchir Balaji, Shantanu Jain, William Saunders, Christopher Hesse, Andrew N. Carr, Jan Leike, Josh Achiam, Vedant Misra, Evan Morikawa, Alec Radford, Matthew Knight, Miles Brundage, Mira Murati, Katie Mayer, Peter Welinder, Bob McGrew, Dario Amodei, Sam McCandlish, Ilya Sutskever, and Wojciech Zaremba. Evaluating large language models trained on code, 2021. URL https://arxiv.org/abs/2107.03374.

Yujia Li, David Choi, Junyoung Chung, Nate Kushman, Julian Schrittwieser, Rémi Leblond, Tom Eccles, James Keeling, Felix Gimeno, Agustin Dal Lago, Thomas Hubert, Peter Choy, Cyprien de Masson d'Autume, Igor Babuschkin, Xinyun Chen, Po-Sen Huang, Johannes Welbl, Sven Gowal, Alexey Cherepanov, James Molloy, Daniel J. Mankowitz, Esme Sutherland Robson, Pushmeet Kohli, Nando de Freitas, Koray Kavukcuoglu, and Oriol Vinyals. Competition-level code generation with alphacode. *Science*, 378(6624):1092–1097, December 2022. ISSN 1095-9203. doi: 10.1126/science.abq1158. URL http://dx.doi.org/10.1126/science.abq1158.

Stack Overflow Developer Survey Team. 2024 developer survey: Most popular, admired, and desired technologies. https://survey.stackoverflow.co/2024/technology, 2024. Accessed: 2025-06-12.

Stack Overflow. 2022 developer survey. https://survey.stackoverflow.co/2022, 2022. Accessed June 2025.

Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, Victor Zhong, Caiming Xiong, Ruoxi Sun, Qian Liu, Sida Wang, and Tao Yu. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows, 2025. URL https://arxiv.org/abs/2411.07763.

Mohammadreza Pourreza and Davood Rafiei. Din-sql: Decomposed in-context learning of text-to-sql with self-correction, 2023. URL https://arxiv.org/abs/2304.11015.

Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. Codes: Towards building open-source language models for text-to-sql, 2024. URL https://arxiv.org/abs/2402.16347.

Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. Chess: Contextual harnessing for efficient sql synthesis, 2024. URL https://arxiv.org/abs/2405.16755.

Zijin Hong, Zheng Yuan, Hao Chen, Qinggang Zhang, Feiran Huang, and Xiao Huang. Knowledge-to-sql: Enhancing sql generation with data expert llm, 2024. URL https://arxiv.org/abs/2402.11517.

Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, Zilin Zhang, and Dragomir Radev. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task. In Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, editors, *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 3911–3921, Brussels, Belgium, October-November 2018. Association for Computational Linguistics. doi: 10.18653/v1/D18-1425. URL https://aclanthology.org/D18-1425/.

Victor Zhong, Caiming Xiong, and Richard Socher. Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, abs/1709.00103, 2017.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs, November 2023a. URL http://arxiv.org/abs/2305.03111. arXiv:2305.03111 [cs].

Minghang Deng, Ashwin Ramachandran, Canwen Xu, Lanxiang Hu, Zhewei Yao, Anupam Datta, and Hao Zhang. Reforce: A text-to-sql agent with self-refinement, consensus enforcement, and column exploration, 2025.

Juan Sequeda, Dean Allemang, and Bryon Jacob. A benchmark to understand the role of knowledge graphs on large language model's accuracy for question answering on enterprise sql databases, 2023. URL https://arxiv.org/abs/2311.07509.

Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Rongyu Cao, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin C. C. Chang, Fei Huang, Reynold Cheng, and Yongbin Li. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls, 2023b. URL https://arxiv.org/abs/2305.03111.