

Neurosymbolic Join Resolution for Enterprise Text-to-SQL

Silvan Wiedmer¹, Albert Weichselbraun^{1,2,*}, Adrian M.P. Braşoveanu³ and Norman Süsstrunk¹

¹University of Applied Sciences of the Grisons, 7000 Chur, Switzerland

²webLyzard technology, 1090 Vienna, Austria

³Modul University Vienna, 1190 Vienna, Austria

Abstract

Enterprise Text-to-SQL systems enable natural-language access to relational databases but remain challenging in real-world environments, where large schemas, undocumented business semantics, and implicit relationships frequently lead to incorrect SQL joins. These limitations are particularly problematic in high-stakes domains requiring reliable, explainable, and on-premises AI solutions. This paper presents a neurosymbolic approach that combines lightweight language models with a symbolic Semantic Join Path Resolver operating on a schema-level knowledge graph. Rather than requiring the language model to infer complex join sequences, the symbolic component deterministically computes valid join paths, allowing the model to focus on semantic interpretation and generation of the remaining SQL clauses. We evaluate the approach on an enterprise benchmark derived from a production database containing nearly 300 tables, comparing it with a purely neural Text-to-SQL baseline across five open-weight models ranging from 1.7B to 31B parameters. Symbolic join resolution raises execution accuracy substantially for models of at least 8B parameters, holding Soft F1 close to 1 where neural baselines degrade, and allows an 8B model to match a 31B one. Below that capability threshold the effect reverses: the smallest models fail to exploit the resolver’s output and score lower with it than without. The proposed architecture shows both how neurosymbolic AI can improve the reliability of enterprise natural-language interfaces under practical deployment constraints, and where that benefit runs out.

Keywords

Enterprise Data, Knowledge Graph, Natural Language Interface, Neurosymbolic AI, Text2SQL

1. Introduction

Natural language interfaces promise to make enterprise information systems accessible to non-technical users, but in regulated domains such as wealth management and healthcare incorrect answers are unacceptable, and deployed systems must additionally satisfy requirements for data security, explainability, and on-premise execution that preclude frontier cloud-hosted models. A central obstacle in this setting is the construction of correct SQL joins. Production databases contain hundreds of tables, undocumented business logic and semantic relationships that no foreign-key constraint records, so language models frequently fail to infer the correct join sequence even once the relevant tables have been identified. Although benchmarks such as BIRD and Spider 2 have substantially advanced the field, state-of-the-art systems remain well short of human experts on real enterprise schemas [1, 2], and studies on industrial datasets identify incorrect join generation as one of the dominant failure modes [3].

Existing remedies largely address schema linking, retrieval, and decoding. Component-level schema retrieval, graph-based traversal of schema graphs, and supervised schema linking narrow and structure the schema presented to the model [4, 5], while grammar-constrained generation guarantees that the emitted string is syntactically valid SQL [6, 7, 8]. Syntactic validity is not semantic correctness, however: a well-formed query over the wrong join path executes silently and returns a plausible but incorrect

Posters, Demos, Blue Sky, and Tutorials at SEMANTiCS 2026, Sep 2026, Ghent, Belgium

*Corresponding author.

✉ silvan.wiedmer@fhgr.ch (S. Wiedmer); albert.weichselbraun@fhgr.ch (A. Weichselbraun); adrian.brasoveanu@modul.ac.at (A. M.P. Braşoveanu); norman.suesstrunk@fhgr.ch (N. Süsstrunk)

🆔 0009-0006-8760-8749 (S. Wiedmer); 0000-0001-6399-045X (A. Weichselbraun); 0000-0002-3439-4736 (A. M.P. Braşoveanu); 0009-0006-8974-797X (N. Süsstrunk)



© 2026 Copyright for this paper by its authors. Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

result, which is precisely the failure enterprise users cannot detect. It has also been argued that explicit schema linking becomes unnecessary as models improve [9]; our results qualify that position, since the benefit of symbolic support turns out to depend sharply on model capacity rather than to vanish with it. What these approaches share is the assumption that valid joins can be recovered from the database schema itself, which does not hold where business relationships are encoded only implicitly.

A second line of work grounds language models in enterprise semantics through knowledge graphs, showing that schema-level ontologies supply the business concepts and relationships that relational schemas omit [3, 10, 11]. Classical ontology-based data access exposes relational data as RDF through declarative mappings [12, 13]; we adopt the mapping idea without the exposure, as our graph describes schema semantics only, is never queried by end users, and serves exclusively as symbolic background knowledge for SQL generation. This design sits within the broader neurosymbolic programme of pairing learned components with explicit symbolic structure [14, 15].

This paper contributes the design and evaluation of a symbolic *Semantic Join Path Resolver* that computes valid join sequences over such a schema-level knowledge graph, leaving a lightweight language model to generate the remaining query components. Confining probabilistic reasoning to semantic interpretation provides deterministic guarantees at the most error-prone stage of enterprise SQL generation and lowers the reasoning burden on small on-premise models. We evaluate the resolver on a systematic join benchmark covering join paths of one to seven joins, across five open-weight models spanning 1.7B to 31B parameters.

2. Method

Our approach assigns complementary roles to neural and symbolic components: the language model interprets the user’s request and identifies the relevant business concepts and corresponding database tables, while deterministic symbolic components construct the executable SQL. Figure 1 illustrates the workflow. The identified concepts and their corresponding database tables are provided as input to the *Semantic Join Path Resolver*, which deterministically computes a valid join path and returns a SQL join statement connecting the required tables. The language model then uses this join statement to generate the remaining clauses (projection, filtering, and aggregation) before the query is executed.

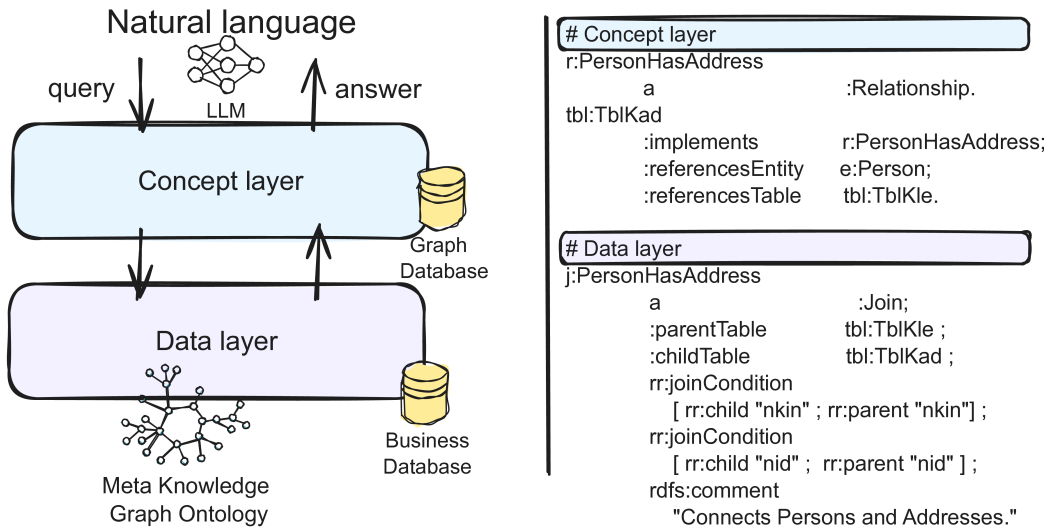


Figure 1: Overview of the proposed neurosymbolic query pipeline. Symbolic reasoning is responsible for deterministic join construction, while the language model performs semantic interpretation and SQL synthesis (left). RDF snippet illustrating the linking between conceptual relationships and executable semantic join definitions (right).

The resolver operates on a lightweight schema-level knowledge graph that contains no business

instances and is never queried directly by end users; it serves purely as symbolic background knowledge. Its *concept layer* models business concepts independently of their technical implementation, while the *data layer* represents executable database structures including tables, attributes and explicitly modelled semantic joins, and links between the layers translate conceptual requests into executable SQL. The graph is built semi-automatically from database metadata, technical documentation, and manually curated join definitions, using deterministic extraction for structural information and language models only to link technical artefacts to business concepts. Because it holds schema metadata alone, it remains stable under ordinary data updates and is regenerated only when the schema changes.

Enterprise databases connect business entities through bridge tables, composite keys and application-specific conventions that foreign-key constraints do not record, which is why language models generate incorrect joins even when the relevant tables have been identified. The resolver addresses this by associating each semantic relationship in the graph with one or more join definitions specifying the participating tables and join conditions. Given the tables identified by the language model, the resolver traverses the graph to find a valid connection between them using these modelled relationships rather than foreign keys alone; the resulting join sequence is deterministic and independent of the language model. The RDF snippet in Figure 1 shows how conceptual relationships are linked to these executable join definitions.

The computed join path is supplied to the language model as structured context, so the model generates only projections, predicates and aggregation operators. Throughout the evaluation we employ Qwen3-1.7B, Llama 3.2-3B, Qwen3-8B-FP8, Ministral-3-8B and Gemma-4-31B-it, served using vLLM without fine-tuning. This separation confines probabilistic reasoning to semantic interpretation, substantially reducing the complexity of the generation task and allowing comparatively small on-premise models to produce reliable SQL over complex enterprise schemas.

3. Evaluation

The evaluation investigates whether symbolic join reasoning improves SQL generation compared to a purely neural approach across different models. Our experiments focus exclusively on SQL join construction rather than end-to-end question answering or knowledge graph construction to isolate the impact of this component from other effects.

3.1. Experimental Setup

The benchmark compares a purely neural Text-to-SQL pipeline with the proposed neurosymbolic approach. In both settings, the language model receives the relevant subset of the database schema and is asked to generate an executable SQL query connecting two specified tables and return it using a tool call.

In the neural baseline the model must infer the complete join sequence from the schema alone; in the neurosymbolic setting it may additionally call the *Semantic Join Path Resolver*, whose returned join sequence enters the context before generation, leaving the model to produce only projections and filtering conditions.

The dataset is derived from the semantic join definitions in the knowledge graph by automatically enumerating all valid join paths of up to seven joins, yielding 216 questions of increasing structural complexity; Table 1 gives the distribution of required join lengths.

Table 1

Join table distribution in the dataset

Required number of joins	1	2	3	4	5	6	7
Count	36	44	42	38	32	16	8

To assess the model independence of the approach, the benchmark is run across several open-weight

models; results are reported for Qwen3-1.7B, Llama 3.2-3B, Qwen3-8B-FP8, Ministral-3-8B and Gemma-4-31B-it, spanning 1.7B to 31B parameters. Each question is executed four times to account for stochastic generation, with up to three retries on invalid tool calls or unexecutable SQL, and results are compared against the reference queries using the Soft F1 score of the BIRD-SQL Mini-Dev benchmark.

3.2. Results

Table 2 reports Soft F1 statistics aggregated over the full benchmark. As expected, the purely neural approach benefits from model size and from advances in model architecture and training techniques, with mean scores rising from 0.32 for Llama 3.2-3B to 0.77 for Gemma-4-31B-it. Neural performance is also highly dispersed: first quartiles at or near zero for three of the five models reflect frequent complete failures, which a breakdown by join length shows to be concentrated on the longer join paths. The neurosymbolic approach does not only raise overall performance but also makes it consistent, yielding a mean Soft F1 close to 1 with standard deviations below 0.05 for the two strongest models. Interestingly, this holds only for models of at least 8 billion parameters. The two smaller models, Qwen3-1.7B and Llama 3.2-3B, do not benefit from the join tool results, with their performance decreasing when compared to the neural approach. This indicates that the resulting tool answer is still too complex for the models.

Table 2

Average Table Join Benchmark Scores comparing the Neural and Neurosymbolic approach on 216 test questions with 4 runs per question, aggregated over all join lengths. The scores in bold indicate the best scores per measurement; for Std, lower is better.

Approach	Model	Mean	Std	Min	Q1	Median	Q3	Max
Neural	Qwen3-1.7B	0.5450	0.4120	0.0000	0.0000	0.7240	0.8860	1.0000
Neural	llama3.2-3B	0.3200	0.4170	0.0000	0.0000	0.0000	0.8090	1.0000
Neural	Qwen3-8B-FP8	0.5960	0.3830	0.0000	0.0060	0.7370	0.8900	1.0000
Neural	Ministral-3-8B-2512	0.7210	0.3260	0.0000	0.6820	0.8450	0.9420	1.0000
Neural	gemma-4-31B-it	0.7700	0.2620	0.0000	0.6950	0.8370	0.9740	1.0000
Neurosymbolic	Qwen3-1.7B	0.4820	0.4780	0.0000	0.0000	0.5540	1.0000	1.0000
Neurosymbolic	llama3.2-3B	0.0650	0.2300	0.0000	0.0000	0.0000	0.0000	1.0000
Neurosymbolic	Qwen3-8B-FP8	0.9970	0.0360	0.0390	1.0000	1.0000	1.0000	1.0000
Neurosymbolic	Ministral-3-8B-2512	0.9460	0.1880	0.0000	1.0000	1.0000	1.0000	1.0000
Neurosymbolic	gemma-4-31B-it	0.9950	0.0440	0.0000	1.0000	1.0000	1.0000	1.0000

A qualitative analysis of the failed neural approach traces show that the models often join columns that can be joined based on the used data types but are not intended to semantically, or only joining with a subset of the required columns. This indicates that the language models struggle to infer semantic relationships from schema information alone.

When the neurosymbolic approach fails, the failures are concentrated primarily in the smaller models, Qwen3-1.7B and Llama 3.2-3B, which frequently fail to use the provided join path in its entirety or omit it altogether. This reflects the current prototype’s reliance on the language model to translate the user’s natural language query into the semantic concepts and data entities required by the join path resolver. Smaller models appear less capable of reliably performing this translation and subsequently incorporating the resulting structured join information into the generated query. The observed failures therefore reflect limitations in the model’s ability to use the symbolic output rather than limitations of the symbolic join algorithm itself.

Comparing Qwen3-8B-FP8 with Ministral-3-8B and Gemma-4-31B-it, both of the latter clearly outperform Qwen in the neural setting. Once the join tool is introduced this ordering disappears: Qwen3-8B-FP8 (0.9970) and Gemma-4-31B-it (0.9950) become indistinguishable, and Ministral-3-8B reaches a comparable mean (0.9460) with a noticeably wider spread.

3.3. Discussion

Two findings deserve emphasis. First, symbolic join support is not uniformly beneficial: it requires a capability floor. Below roughly 8B parameters, our experiments indicate that models increasingly struggle to correctly interpret and integrate the resolver’s structured output into the generated query. Qwen3-1.7B and Llama 3.2-3B frequently fail to incorporate the provided semantic join path correctly and consequently even score lower with the resolver than without it. Neurosymbolic decomposition therefore cannot fully compensate the limitations of arbitrarily small models, and reported gains should always be qualified by the capacity of the model receiving the symbolic input.

Second, above that floor the symbolic component substitutes for parameter count. Gemma-4-31B-it holds a substantial neural-baseline advantage over Qwen3-8B-FP8 that disappears once both models have access to the resolver. For on-premise deployment under fixed hardware this is the more consequential result than the aggregate improvement: an 8B model with symbolic join support is a viable alternative to a 31B model without it, at markedly lower memory, latency, and energy cost.

Three limitations qualify these results. The benchmark is enumerated from the same semantic join definitions the resolver consults, so it measures whether the model exploits correct join information rather than whether those definitions generalise to unseen relationships; validating the curated definitions themselves is a separate problem. Soft F1 compares result sets rather than query semantics, so a query may score highly through coincidental overlap. Finally, all measurements come from a single production database of nearly 300 tables in one domain, and the capability threshold in particular may shift with schema size and model family.

4. Conclusions and Outlook

This paper presents a neurosymbolic approach to enterprise Text-to-SQL that pairs a lightweight Large Language Model (LLM) with a schema-level knowledge graph and deterministic semantic join resolution. Delegating join construction to a symbolic resolver substantially improves execution accuracy from 8B parameters upward and removes the degradation that purely neural generation exhibits as join complexity grows, holding Soft F1 close to 1 across join lengths where neural baselines fall away. This targets one of the principal failure modes reported for enterprise Text-to-SQL directly: rather than requiring the language model to infer valid joins from the relational schema, the knowledge graph provides the required join paths, allowing the model to focus on generating projections, predicates, and aggregation logic.

The practical consequence for regulated on-premise settings is that reliability can be bought through symbolic structure rather than through model size, which matters where hardware and data residency are fixed constraints. That substitution has a floor, however, and identifying where it lies for a given schema is a prerequisite for deployment rather than a detail.

The prototype still relies on the language model to translate a natural language query into the concepts and data entities the resolver consumes, which is where the observed failures originate. Future work will therefore replace this step with a fully symbolic alignment layer, extend the knowledge graph with formal constraints for validating generated queries and results, and evaluate the approach end-to-end on a curated use-case benchmark and on additional enterprise schemas.

Declaration on Generative AI

The authors used ChatGPT and LanguageTool for content enhancement, to paraphrase and reword, improve writing style, and for grammar and spelling checks. All outputs were subsequently reviewed and edited by the authors, who take full responsibility for the content of the publication.

References

- [1] J. Li, B. Hui, G. Qu, J. Yang, B. Li, B. Li, B. Wang, B. Qin, R. Geng, N. Huo, X. Zhou, C. Ma, G. Li, K. C.-C. Chang, F. Huang, R. Cheng, Y. Li, Can LLM already serve as a database interface? A BIG bench for large-scale database grounded text-to-SQLs, in: *Advances in Neural Information Processing Systems (NeurIPS), Datasets and Benchmarks Track*, volume 36, 2023. [arXiv:2305.03111](#).
- [2] P. Wang, B. Sun, X. Dong, Y. Dai, H. Yuan, M. Chu, Y. Gao, X. Qi, P. Zhang, Y. Yan, Agentar-scale-SQL: Advancing text-to-SQL through orchestrated test-time scaling, *arXiv preprint arXiv:2509.24403* (2025). doi:10.48550/arXiv.2509.24403.
- [3] J. Sequeda, D. Allemang, B. Jacob, A benchmark to understand the role of knowledge graphs on large language model's accuracy for question answering on enterprise SQL databases, in: *Proceedings of the 7th Joint Workshop on Graph Data Management Experiences & Systems (GRADES) and Network Data Analytics (NDA)*, ACM, 2024, pp. 1–12. doi:10.1145/3661304.3661901.
- [4] J. Eben, A. Ahmad, S. Lau, RASL: Retrieval augmented schema linking for massive database text-to-SQL, *arXiv preprint arXiv:2507.23104* (2025). doi:10.48550/arXiv.2507.23104.
- [5] A. Safdarian, M. Mohammadi, E. J. Bashirloo, M. S. Naderi, H. Faili, SchemaGraphSQL: Efficient schema linking with pathfinding graph algorithms for text-to-SQL on large-scale databases, in: *Findings of the Association for Computational Linguistics: EACL 2026*, Association for Computational Linguistics, Rabat, Morocco, 2026, pp. 2585–2599. doi:10.18653/v1/2026.findings-eacl.134.
- [6] T. Scholak, N. Schucher, D. Bahdanau, PICARD: Parsing incrementally for constrained autoregressive decoding from language models, in: *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2021, pp. 9895–9901. doi:10.18653/v1/2021.emnlp-main.779.
- [7] S. Geng, M. Josifoski, M. Peyrard, R. West, Grammar-constrained decoding for structured NLP tasks without finetuning, in: *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2023, pp. 10932–10952. doi:10.18653/v1/2023.emnlp-main.674.
- [8] Y. Dong, C. F. Ruan, Y. Cai, Z. Xu, Y. Zhao, R. Lai, T. Chen, XGrammar: Flexible and efficient structured generation engine for large language models, in: *Proceedings of Machine Learning and Systems (MLSys)*, volume 7, 2025. doi:10.48550/arXiv.2411.15100.
- [9] K. Maamari, F. Abubaker, D. Jaroslawicz, A. Mhedhbi, The death of schema linking? Text-to-SQL in the age of well-reasoned language models, *arXiv preprint arXiv:2408.07702* (2024). doi:10.48550/arXiv.2408.07702.
- [10] D. Allemang, J. Sequeda, Increasing the accuracy of LLM question-answering systems with ontologies, in: *The Semantic Web – ISWC 2024*, volume 15233 of *Lecture Notes in Computer Science*, Springer, Cham, 2025, pp. 324–339. doi:10.1007/978-3-031-77847-6_18.
- [11] C. Civili, E. Sherkhonov, R. E. K. Stirewalt, Using off-the-shelf LLMs to query enterprise data by progressively revealing ontologies, in: *The Semantic Web: ESWC 2025 Satellite Events*, volume 15832 of *Lecture Notes in Computer Science*, Springer, Cham, 2026. doi:10.1007/978-3-031-99554-5_46.
- [12] S. Das, S. Sundara, R. Cyganiak, R2RML: RDB to RDF mapping language, W3C Recommendation, 2012. URL: <https://www.w3.org/TR/r2rml/>.
- [13] F. Priyatna, O. Corcho, J. Sequeda, Formalisation and experiences of R2RML-based SPARQL to SQL query translation using morph, in: *Proceedings of the 23rd International Conference on World Wide Web (WWW)*, 2014, pp. 479–490. doi:10.1145/2566486.2567981.
- [14] A. d'Avila Garcez, L. C. Lamb, Neurosymbolic AI: The 3rd wave, *Artificial Intelligence Review* 56 (2023) 12387–12406. doi:10.1007/s10462-023-10448-w.
- [15] L. N. DeLong, R. F. Mir, J. D. Fleuriot, Neurosymbolic AI for reasoning over knowledge graphs: A survey, *IEEE Transactions on Neural Networks and Learning Systems* 36 (2024) 7822–7842. doi:10.1109/TNNLS.2024.3420218. [arXiv:2302.07200](#).