How Far Can They Map? Probing LLM Capabilities for Cross-Schema SQL Generation

Mohammadreza Daviran
University of Alberta
Edmonton, AB, Canada
daviran@ualberta.ca

Davood Rafiei
University of Alberta
Edmonton, AB, Canada
drafiei@ualberta.ca

***Abstract*—We study the emerging task of cross-schema SQL mapping, where large language models (LLMs) translate SQL queries across databases with different schemas. This task tests whether models can preserve source query intent and structural properties while adapting to unfamiliar target schema elements. To understand the sources of difficulty, we analyze a dataset of over 100K mapped queries from SQL-Exchange, spanning two benchmarks and two LLMs. We define a taxonomy of more than 50 structural and schema-related features and identify three key signals—query complexity, schema mismatch, and mapping flexibility—that strongly influence execution correctness and semantic validity. Our analysis reveals how different forms of source-target schema mismatch drive model errors and highlights concrete directions for building more robust schema-schema mapping systems.**

***Index Terms*—SQL translation, Schema mapping, Large language models**

I. Introduction

LLMs have shown remarkable capabilities in translating natural language questions into executable SQL queries [1]. Recent advances have enabled these models to generate complex queries over unseen databases by leveraging sophisticated in-context learning and reasoning frameworks [2]–[4]. However, a critical challenge remains underexplored: Can these models robustly transfer queries across schemas with different structures and vocabularies?

Cross-schema SQL mapping refers to the task of translating a SQL query written over a source schema into an *equivalent query* over a target schema, such that the structure and semantics of the original query is preserved while adapting it to a new schema that may differ significantly in table layout, naming conventions, and attribute organization. This capability is particularly useful for generating query workloads for databases that lack existing queries—such as creating schema-specific NL–SQL training data, bootstrapping query repositories, or constructing benchmarks—by mapping queries from other schemas [5]. It can also support adapting queries across heterogeneous schemas in scenarios such as schema evolution, federated query environments, and data integration.

While recent work has leveraged LLMs for schema matching and mapping [6], [7] and SQL translation across dialects [8], [9], these approaches typically assume identical or nearly aligned schemas, and do not address structural translation between semantically divergent databases.

In this work, we conduct a detailed empirical study of LLM performance on cross-schema SQL mapping using SQL-Exchange [5], a recent framework that automatically generates large-scale mapping examples from prompting LLMs on benchmark datasets. By analyzing more than 100K mapping instances generated by SQL-Exchange, we investigate which factors make mappings succeed or fail.

We define a taxonomy of over 50 static features characterizing each mapping instance, grouped into three categories: *Query Complexity*, *Schema Feasibility*, and *Mapping Flexibility*. These features quantify the structural difficulty of the query, the compatibility between source and target schemas, and the availability of mapping paths in the target schema. Our analysis focuses on how these features correlate with three key evaluation metrics: *Execution Validity*, *Semantic Validity*, and *Structural Alignment*. Across all model and benchmark combinations, we observe consistent trends: queries with higher structural complexity or lower schema compatibility are significantly more prone to failure across all metrics. Additionally, mappings with minimal table alignment options tend to reduce success rates, though too many options can introduce ambiguity in table selection.

This study provides a fine-grained analysis of structural factors influencing LLM performance in cross-schema mappings. By linking performance to interpretable static features, our findings inform future work on prompt design, schema conditioning, and evaluation of LLMs in multi-schema settings.

II. Setup

To study the behavior of LLMs in cross-schema SQL mapping tasks, we utilize the outputs of the SQL-Exchange framework. This framework provides a large-scale corpus of over 100,000 automatically generated SQL query mappings between different database schemas. These mappings were produced by prompting two major LLM families—GPT (specifically, GPT-4o mini) and Gemini (Gemini-1.5-flash)—on two widely-used SQL benchmarks: Spider [10] and BIRD [11].

In the SQL-Exchange framework, each LLM was independently prompted to map training queries from both benchmarks to the respective development schemas by preserving the original SQL structure while replacing schema-specific elements—tables, columns, and constants—with corresponding

elements in the target schema. This process results in four distinct experimental settings: BIRD–GPT, BIRD–Gemini, Spider–GPT, and Spider–Gemini, which are analyzed separately.

To evaluate the quality of the generated mappings, we adopt three metrics defined by SQL-Exchange: **Structural Alignment**, **Execution Validity**, and **Semantic Validity**. Structural Alignment captures whether the predicted query preserves the key structural elements (e.g., clauses and logical flow) of the source query. Execution Validity checks whether the generated query runs successfully on the target database, reflecting syntactic and runtime correctness. Semantic Validity evaluates whether the predicted SQL query is meaningful with respect to the target schema and semantically consistent with its natural language description. This is assessed by generating a natural language question from the SQL and verifying that it accurately reflects the intent of the query.

III. A Taxonomy of Mapping Features

To analyze the structural factors that influence LLM performance in cross-schema SQL mapping, a diverse set of over 50 atomic features is examined for each mapping instance. These features are derived through static analysis of the source SQL query, the source and target schemas, and their relational compatibility. Each feature is computed automatically and captures a specific aspect of task difficulty, feasibility, or ambiguity inherent to the mapping process.

The feature design is guided by the hypothesis that query-level and schema-level characteristics impose distinct constraints on mapping success. To organize this feature space, the features are grouped into three high-level categories: Query Complexity, Schema Feasibility, and Mapping Flexibility.

A. Query Complexity ($\mathcal{C}$)

This group captures the intrinsic complexity of the source SQL query—irrespective of the target schema. Features in this category describe the syntactic and logical structure of the input query, including its use of joins, filtering predicates, grouping logic, nesting depth, and literal values. Queries with deeper logical composition or broader surface area pose a greater burden on the model to preserve structure and semantics during transfer.

For instance, features like the number of joins and number of WHERE conditions reflect the structural components that must be reconstructed in the target schema, while the number of literals introduces challenges for natural language generation and semantic grounding of the mapped SQL query. To summarize the overall logical burden of a query, we adopt the size of the abstract syntax tree (AST) as the key indicator for this group. This feature measures the total number of nodes in the query's abstract syntax tree, capturing the syntactic depth and breadth of the query. Larger trees indicate more compositional logic, deeper nesting, and increased difficulty in preserving the original intent during schema adaptation.

B. Schema Feasibility ($\mathcal{F}$)

These features evaluates whether the requirements of a source query can be realistically satisfied by the target schema. Even simple queries may fail if essential schema elements such as tables or columns do not exist in the target. These features quantify the degree of mismatch in table availability, foreign key coverage, and especially, column-level compatibility.

As the representative feature for this group, we select the number of source columns without a compatible target column. This feature counts how many attributes referenced in the source query lack a type-compatible counterpart in the target schema. It directly captures fundamental attribute-level incompatibility, and was consistently one of the strongest predictors of mapping failure. A high value suggests a structural dead end, where the model is forced to hallucinate or rewrite the query in ways that undermine semantic integrity.

C. Mapping Flexibility ($\mathcal{M}$)

Mapping Flexibility captures the structural degrees of freedom available when aligning elements from the source query to the target schema. Even when a mapping is technically feasible, multiple valid alignment alternatives may exist—for example, several target tables that could plausibly fulfill the role of a single source table. Such flexibility can facilitate adaptation by offering alternative structural correspondences, particularly when schemas differ in organization or granularity. However, when alignment choices become overly numerous or insufficiently constrained, the decision space may become unstable, increasing the risk of semantic drift.

This group includes features that characterize the size and redundancy of the target schema and the availability of viable mapping paths. While a richer schema may offer more opportunities for alignment, excessive flexibility can lead to indecision or semantic drift. Among these features, the minimum number of compatible target tables per source table is especially informative. It captures bottlenecks in the mapping process, where at least one source table has few or no viable alignment candidates. Low values suggest structural sparsity that constrains mapping options, while very high values may introduce excessive ambiguity. Both extremes elevate the risk of inaccurate translation.

IV. Empirical Results

To better understand how input query and schema mapping characteristics affect model performance, we analyze the main features introduced in Section III, focusing on their correlation with key evaluation metrics across experimental settings.

In cross-schema SQL mapping, LLMs are expected to preserve both the structure and intent of the source SQL query when adapting it to a new schema. However, as noted in SQL-Exchange, models, especially Gemini-1.5-flash, sometimes deviate from this goal by simplifying or restructuring the source query, potentially compromising the mapping accuracy. This behavior occurs in 13.2% to 33.4% of cases, depending on the benchmark and model [5].

To isolate true mapping difficulty from such generation-level artifacts, we first focus on predictions that preserve the source query structure, enabling a cleaner study of semantic failure modes. Second, we study how feature conditions affect the model's decision to change or preserve the query structure.

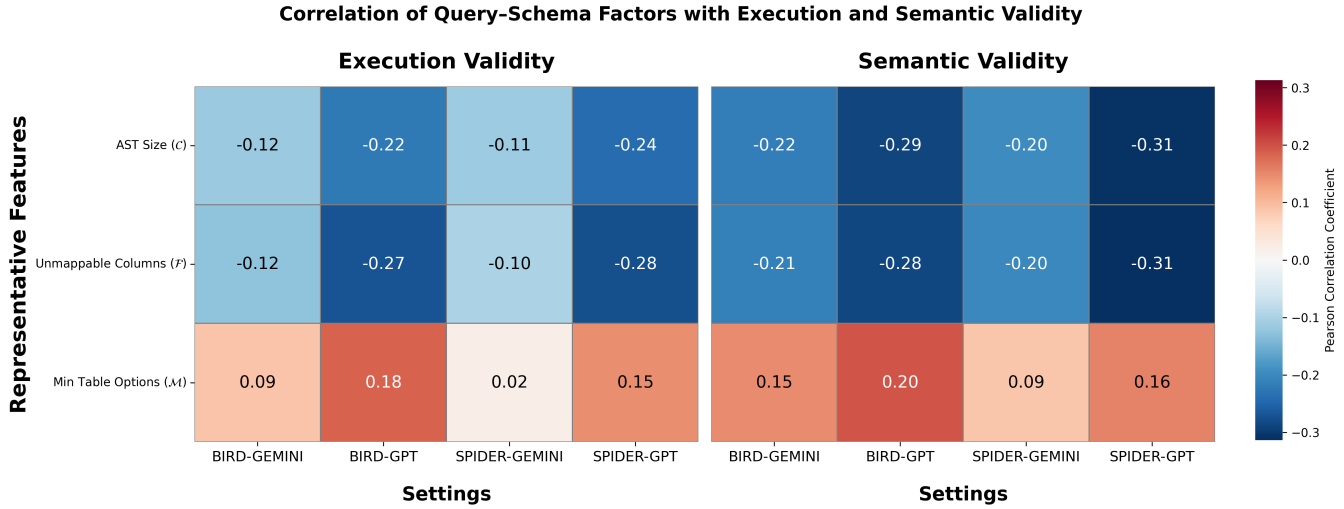

Fig. 1. Correlation heatmap between representative features and Semantic and Execution Validity for template-preserving queries across settings. All observed correlations are statistically significant ($p < 0.01$).

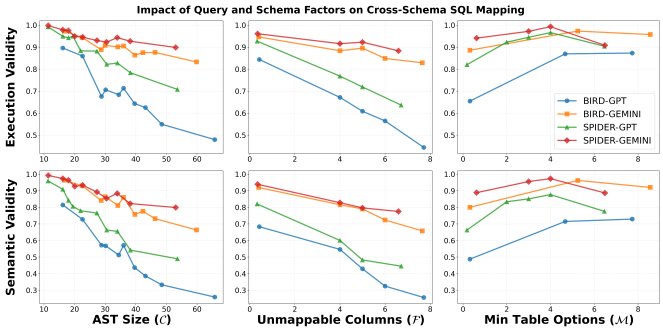

Fig. 2. Impact of representative features on execution and semantic outcomes across settings.

A. Impact on Execution and Semantic Validity

Figure 1 presents Pearson correlation coefficients between key structural features—AST size, number of unmappable source columns, and minimum table mapping options—and model performance (Execution and Semantic Validity), while Figure 2 illustrates outcome trends across different feature values. Together, these reveal how different query and schema characteristics influence cross-schema mapping outcomes.

a) Query Complexity: We observe a consistently negative correlation between query structural complexity (measured by AST size) and both Execution Validity and semantic validity. The correlation is especially strong in the Spider-GPT setting (–0.24 and –0.31 for execution and semantic metrics, respectively), where deeper query structures tend to increase the risk of failure. This suggests that deeper nesting and compositional logic significantly increase the cognitive burden for models, particularly when they must preserve the original structure. Gemini appears more robust to complex queries than GPT in both benchmarks, yet the decline in quality with rising AST size remains visible across all settings.

b) Schema Feasibility: The number of unmappable source columns emerges as a strong negative predictor of performance. It exhibits among the highest absolute correlations with both metrics, especially in BIRD-GPT and SPIDER-GPT (–0.27 and –0.28 for Execution Validity; –0.28 and –0.31 for semantic validity). This trend confirms that schema-level incompatibilities pose a hard constraint on the success of query mapping. Importantly, even small increases in this feature lead to sharp performance drops (Figure 2, center), marking it as a key bottleneck in cross-schema transfer.

c) Mapping Flexibility: The minimum number of compatible target tables for source tables has a mild positive correlation with both metrics across all configurations (e.g., 0.15 and 0.20 for Semantic Validity in BIRD-Gemini and BIRD-GPT). This suggests that having at least one mapping path per source table improves success rate, although the benefit saturates. As shown in figure 2, too many options can backfire by increasing model confusion, reflecting the delicate balance between flexibility and ambiguity.

Beyond the primary feature groups, certain surface-level schema characteristics also impact execution outcomes. In the BIRD benchmark, which contains many column names with spaces, we observe a meaningful negative correlation between the proportion of such columns and Execution Validity (–0.21 in BIRD–Gemini, –0.12 in BIRD–GPT; both $p < 0.01$). These failures typically occur when LLMs omit or mishandle quoting for non-standard column names, producing incorrect references. This pattern does not appear in Spider, where column names are generally simpler and less likely to include special characters. These results align with SQL-Exchange's analysis, which identified column reference errors as a common cause of execution failures in complex or noisy schemas.

Together, these results confirm that query structure, schema compatibility, and mapping flexibility jointly influence mapping success.

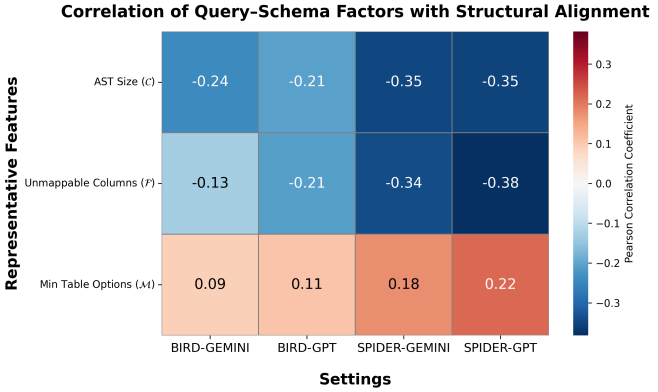

Fig. 3. Correlation heatmap of representative features with Structural Alignment across all queries; all correlations are significant ($p < 0.01$).

B. Impact on Structural Alignment

We now examine what factors influence whether models preserve the structural skeleton of the source query. Unlike the previous metrics, which assess performance conditional on structure being maintained, the *Structural Alignment* metric directly measures if the generated query retains the source query's high-level form.

Figure 3 reports the correlations between our three representative features and structural alignment across all four settings. This analysis includes all predictions—both structure-preserving and not—allowing us to assess what drives the model to alter or maintain the original query form.

Query complexity, measured by AST size, shows a consistently strong negative correlation with structural alignment (–0.35 to -0.21). This indicates that as query depth and logical composition increase, models are more likely to simplify or restructure the query rather than preserve it. The difficulty of retaining deep structure under schema shift is particularly evident in Spider, where the effect is strongest.

The number of unmappable source columns also shows strong negative correlations with structural alignment, particularly for Spider (–0.34 for Gemini, –0.38 for GPT). This confirms that schema-level incompatibilities not only hinder semantic and execution success but also prompt the model to abandon the original structure altogether. When the target schema cannot satisfy even basic column mappings, LLMs tend to reframe or simplify the query, likely in an effort to produce a minimally valid alternative.

In contrast, minimum table mapping options—our proxy for mapping flexibility—show a modest positive correlation (0.09 to 0.22). This suggests that when at least one structural path exists for each source table, models are slightly more inclined to retain the original form. However, the effect is limited, highlighting that excessive flexibility does not necessarily improve structure preservation.

Overall, structural alignment depends on both the intrinsic complexity of the query and the feasibility of finding structurally faithful mappings. Models preserve structure most reliably when the query is simple, the schema is compatible, and the mapping space is flexible enough to support alignment without inducing confusion.

V. Discussion

Our analysis reveals consistent patterns in how query and schema characteristics shape LLM performance on cross-schema SQL mapping. While LLMs like Gemini-1.5-flash and GPT-4o mini perform well overall, their performance varies predictably with query complexity, schema feasibility, and mapping flexibility.

Interestingly, both execution and semantic validity degrade more sharply on BIRD compared to Spider when features such as query complexity or number of unmappable columns increase. This suggests that LLMs are more sensitive to schema messiness, denormalization, and naming ambiguity in real-world settings. For example, surface-level schema ambiguity in BIRD pose challenges not captured by traditional benchmarks like Spider. This highlights the need for evaluation across diverse schema environments to accurately assess generalization.

Most predictive features in our analysis, particularly those related to schema feasibility, are lightweight, static, and easily interpretable. These signals can be leveraged to build predictive tools that anticipate mapping difficulty, rerank or filter model outputs, or trigger fallback mechanisms in pipeline systems. They also open avenues for adaptive prompting strategies, where the LLM is guided differently based on the detected complexity or compatibility of the task.

While our analysis covers a large space of mappings and schema-query configurations, it focuses on one-shot LLM behavior using static prompts. Future work could explore dynamic prompt engineering, schema-aware augmentations, or fine-tuning for cross-schema alignment. Additionally, richer annotation of mappings—e.g., partial success, error type, or alignment breakdown—could support deeper understanding.

VI. Conclusion

We present a systematic analysis of LLM performance on cross-schema SQL mapping using the SQL-Exchange dataset across two model families (GPT and Gemini) and two benchmarks (Spider and BIRD). By defining a taxonomy of over 50 static features, we identify three key dimensions that govern mapping outcomes: query complexity, schema feasibility, and mapping flexibility. Our results show that mappings are most likely to succeed when the source query is not overly complex, the target schema offers compatible attributes, and the mapping space is neither overly constrained nor excessively ambiguous. These findings offer a clearer understanding of the structural and semantic limits of current LLMs in multi-schema environments, and outline the core factors that define the boundary between possible and impossible mappings.

Acknowledgment

This research was supported by the Natural Sciences and Engineering Research Council of Canada (NSERC).

REFERENCES

[1] Z. Hong, Z. Yuan, Q. Zhang, H. Chen, J. Dong, F. Huang, and X. Huang, " Next-Generation Database Interfaces: A Survey of LLM-Based Text-to-SQL ," *IEEE Transactions on Knowledge & Data Engineering*, vol. 37, no. 12, pp. 7328–7345, Dec. 2025. [Online]. Available: https://doi.ieeecomputersociety.org/10.1109/TKDE.2025.3609486

[2] M. Pourreza and D. Rafiei, "Din-sql: decomposed in-context learning of text-to-sql with self-correction," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23. Red Hook, NY, USA: Curran Associates Inc., 2023.

[3] D. Gao, H. Wang, Y. Li, X. Sun, Y. Qian, B. Ding, and J. Zhou, "Text-to-sql empowered by large language models: A benchmark evaluation," *Proc. VLDB Endow.*, vol. 17, no. 5, p. 1132–1145, Jan. 2024. [Online]. Available: https://doi.org/10.14778/3641204.3641221

[4] M. Pourreza, S. Talaei, R. Sun, X. Wan, H. Li, A. Mirhoseini, A. Saberi, S. Arik *et al.*, "Reasoning-sql: Reinforcement learning with sql tailored partial rewards for reasoning-enhanced text-to-sql," *arXiv preprint arXiv:2503.23157*, 2025.

[5] M. Daviran, B. Lin, and D. Rafiei, "Sql-exchange: Transforming sql queries across domains," *arXiv preprint arXiv:2508.07087*, 2025.

[6] M. Parciak, B. Vandevoort, F. Neven, L. M. Peeters, and S. Vansummeren, "Schema matching with large language models: an experimental study," in *Proceedings of Workshops at the 50th International Conference on Very Large Data Bases (VLDB 2024)*. Guangzhou, China: VLDB.org, Aug. 2024. [Online]. Available: https://vldb.org/workshops/2024/proceedings/TaDA/TaDA.8.pdf

[7] N. Seedat and M. Van Der Schaar, "Bootstrapping self-improvement of language model programs for zero-shot schema matching," in *Forty-second International Conference on Machine Learning*, 2025.

[8] A. L. Ngom and T. Kraska, "Mallet: Sql dialect translation with llm rule generation," in *Proceedings of the Seventh International Workshop on Exploiting Artificial Intelligence Techniques for Data Management*, ser. aiDM '24. New York, NY, USA: Association for Computing Machinery, 2024. [Online]. Available: https://doi.org/10.1145/3663742.3663973

[9] W. Zhou, Y. Gao, X. Zhou, and G. Li, "Cracking sql barriers: An llm-based dialect translation system," *Proc. ACM Manag. Data*, vol. 3, no. 3, Jun. 2025. [Online]. Available: https://doi.org/10.1145/3725278

[10] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds. Brussels, Belgium: Association for Computational Linguistics, Oct.-Nov. 2018, pp. 3911–3921. [Online]. Available: https://aclanthology.org/D18-1425/

[11] 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. Chang, F. Huang, R. Cheng, and Y. Li, "Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls," in *Proceedings of the 37th International Conference on Neural Information Processing Systems*, ser. NIPS '23. Red Hook, NY, USA: Curran Associates Inc., 2023.