# Text-to-SQL Conversion Strategies: An Enhanced Analysis

**Candidate**: Tahsin Jawwad
**Supervisor**: Dr. Ramon Lawrence
**Course**: COSC 448 – Directed Studies in Computer Science
**Term**: Winter 2024/25 Term 2

## 1. Introduction

Text-to-SQL is the process of converting natural language questions into SQL queries, enabling users to retrieve database information without explicit SQL knowledge. This capability is crucial for database query tools, educational platforms, and various business applications. My Directed Studies project under Dr. Ramon Lawrence specifically aimed at exploring and enhancing Text-to-SQL methodologies, using Generative AI models, primarily GPT-3.5-turbo, and focusing on schema linking and modular prompting strategies to address limitations found in current methods. The goal, as outlined in the course description, was to investigate different Generative AI approaches, replicate existing experiments, and propose improvements by integrating structural and semantic metadata into the translation process.

## 2. Comparative Evaluation and Replication of Text-to-SQL Frameworks[1]

The initial phase of this research involved evaluating recent Text-to-SQL approaches, specifically LangChain, C3, and DIN-based strategies, as detailed in Nascimento et al. (2024), using an Oracle database on the Mondial dataset.

### 2.1 LangChain

LangChain is a modular framework that integrates Large Language Models (LLMs) with natural language processing for tasks like database querying. It offers methods to convert natural language to SQL. One method, SQLQueryChain, extracts database metadata to directly convert natural language questions into SQL queries. As per the original paper, SQLQueryChain performed the best out of all the LangChain models, achieving the best accuracy out of all the strategies tested in the paper for medium complexity queries (85%). It also performed the best out of all the strategies when replicating the findings as seen in Table 1.

The other LangChain strategies are SQLDatabaseSequentialChain, which first determines relevant tables before invoking SQLQueryChain to narrow the search space, and SQLDatabaseAgent, which interacts flexibly with databases using schema awareness and error recovery.

## 2.2 C3-Based Strategies

The C3 approach uses careful prompt engineering with GPT models to handle schema complexity and ambiguity. It employs Clear Prompting (CP) to include only relevant schema information, Calibration with Hints (CH) to mitigate GPT biases, and Consistent Output (CO) to generate multiple reasoning paths and select the most consistent SQL query. This strategy showed the highest accuracy overall (78%) and for complex queries (71%) in the original paper.

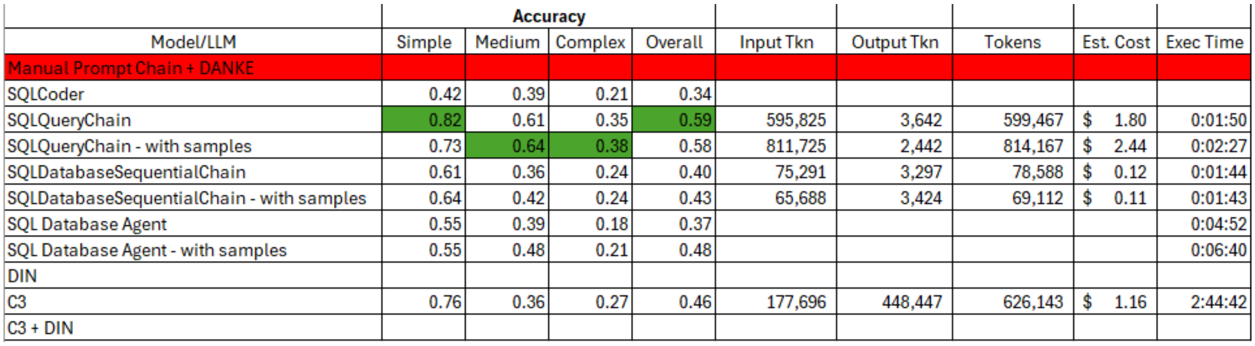
## 2.3 DIN-SQL-Based Strategies

DIN-SQL decomposes the Text-to-SQL conversion into sequential steps: Schema Linking, Query Classification and Decomposition, SQL Generation, and Self-Correction. These steps enhance accuracy and clarity by explicitly matching natural language to schema components, categorizing queries, using intermediate representations, and correcting errors. DIN performed worse than both C3 and C3+DIN despite having a relatively high cost and was not replicated.

## 2.4 Combined Strategies

Nascimento et al. (2024) also explored a combined strategy, "C3+DIN," integrating components from both C3 and DIN-SQL. This approach involves six steps, including schema description using LangChain, clear prompting and schema linking, classification and decomposition, calibration with hints, SQL generation, and self-correction. The original paper found that C3+DIN achieved the highest accuracy on simple queries with GPT-4 (91%). Due to cost, this experiment was not replicated.

## Table 1: Replication of Findings



| | Accuracy | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Model/LLM | Simple | Medium | Complex | Overall | Input Tkn | Output Tkn | Tokens | Est. Cost | Exec Time |
| Manual Prompt Chain + DANKE | | | | | | | | | |
| SQLCoder | 0.42 | 0.39 | 0.21 | 0.34 | | | | | |
| SQLQueryChain | 0.82 | 0.61 | 0.35 | 0.59 | 595,825 | 3,642 | 599,467 | $ 1.80 | 0:01:50 |
| SQLQueryChain - with samples | 0.73 | 0.64 | 0.38 | 0.58 | 811,725 | 2,442 | 814,167 | $ 2.44 | 0:02:27 |
| SQLDatabaseSequentialChain | 0.61 | 0.36 | 0.24 | 0.40 | 75,291 | 3,297 | 78,588 | $ 0.12 | 0:01:44 |
| SQLDatabaseSequentialChain - with samples | 0.64 | 0.42 | 0.24 | 0.43 | 65,688 | 3,424 | 69,112 | $ 0.11 | 0:01:43 |
| SQL Database Agent | 0.55 | 0.39 | 0.18 | 0.37 | | | | | 0:04:52 |
| SQL Database Agent - with samples | 0.55 | 0.48 | 0.21 | 0.48 | | | | | 0:06:40 |
| DIN | | | | | | | | | |
| C3 | 0.76 | 0.36 | 0.27 | 0.46 | 177,696 | 448,447 | 626,143 | $ 1.16 | 2:44:42 |
| C3 + DIN | | | | | | | | | |

## 3. Exploring Advanced Techniques for Improved Accuracy

Beyond the fundamental frameworks, the field of Text-to-SQL includes advanced techniques aimed at enhancing accuracy, especially with complex database schemas.

### 3.1 XiYan-SQL and M-Schema[2,4]

XiYan-SQL is a framework that uses a multi-generator ensemble strategy, including Schema Linking, Candidate Generation, and Candidate Selection, to improve Text-to-SQL conversion. This framework is explored later in this report. A key innovation is M-Schema[5], a semi-structured schema representation that enhances the LLM's understanding of database structures by explicitly showing hierarchical relationships among databases, tables, and columns, including details like data types and primary keys.

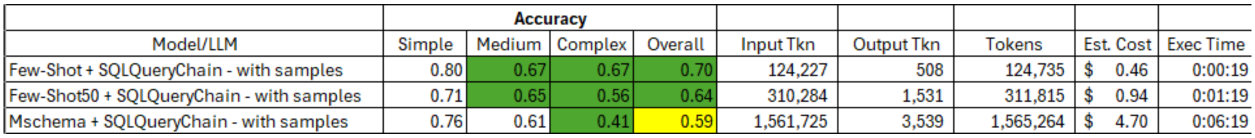
In my research, I focused on implementing the M-Schema representation within the LangChain framework to observe its impact on SQL generation accuracy. I adapted the SQLQueryChain to incorporate M-Schema for the Mondial dataset. The integration of M-Schema resulted in no change in the overall accuracy, but saw an improvement on complex queries as seen in Table 2.

### 3.2 Few-Shot Learning Strategies

Few-Shot Learning in Text-to-SQL involves providing an LLM with a small number of examples (natural language question and corresponding SQL query pairs) to guide the model in translating new questions into SQL.

Given the set of 100 queries in the Mondial dataset, I ran two experiments. The first experiment divided 20 queries into the test set and 80 queries into the training set, from which the top 5 most similar queries, based on cosine similarity, were extracted. The second experiment divided 50 queries into the test set and 50 queries into the training set to increase the size of the test. These experiments were again carried out using SQLQueryChain. Implementing the Few-Short Similar Pairs strategy so an improvement in overall, complex, and medium accuracy in both experiments. However, it is important to note that this could be due to the smaller test set.

### Table 2: Experimentation Results with New Features

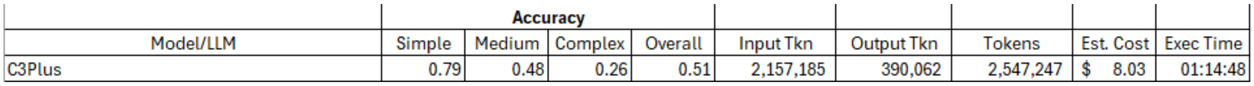


| | Accuracy | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Model/LLM | Simple | Medium | Complex | Overall | Input Tkn | Output Tkn | Tokens | Est. Cost | Exec Time |
| Few-Shot + SQLQueryChain - with samples | 0.80 | 0.67 | 0.67 | 0.70 | 124,227 | 508 | 124,735 | $ 0.46 | 0:00:19 |
| Few-Shot50 + SQLQueryChain - with samples | 0.71 | 0.65 | 0.56 | 0.64 | 310,284 | 1,531 | 311,815 | $ 0.94 | 0:01:19 |
| Mschema + SQLQueryChain - with samples | 0.76 | 0.61 | 0.41 | 0.59 | 1,561,725 | 3,539 | 1,565,264 | $ 4.70 | 0:06:19 |

### 3.3 C3+

The C3+ framework took the provided C3-DIN pipeline and made minor adjustments to it. First, the schema linking code was refactored to incorporate M-Schema. An SQL Validator was also created to catch simple syntax errors in the generated SQL query. Finally, an embedding-based retrieval system was also tested which did not seem to be as accurate as the original schema linking.

### Table 3: Experimentation Results with C3+



| | Accuracy | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Model/LLM | Simple | Medium | Complex | Overall | Input Tkn | Output Tkn | Tokens | Est. Cost | Exec Time |
| C3Plus | 0.79 | 0.48 | 0.26 | 0.51 | 2,157,185 | 390,062 | 2,547,247 | $ 8.03 | 01:14:48 |

The experiment displayed an improvement from the base C3 model but the change was disappointing. This was probably due to the difficulty and inaccurate incorporation of M-Schema, as well as the lack of feedback from the SQL Validator. I continued working on this pipeline, trying to indicate Candidate SQL generation, as seen in XiYan-SQL, and feeding the SQL Validator error messages back into the pipeline to make changes. Other changes include making the extraction of table and column names in the schema linking more robust and using the embeddings-based retrieval system as a fallback. However, this experiment was not run due to the complexity of the changes and lack of time.

## 4. Advanced Text-to-SQL Techniques: Potential Directions for Future Research

### 4.1. XiYan-SQL and M-Schema[3]

XiYan-SQL represents an innovative framework that employs a multi-generator ensemble strategy to improve the accuracy and efficiency of Text-to-SQL conversion. This framework comprises three primary components: Schema Linking, Candidate Generation, and Candidate Selection. A key innovation within XiYan-SQL is the introduction of M-Schema, a semi-structured schema representation method. M-Schema is designed to enhance the Large Language Model's (LLM) understanding of database structures by explicitly illustrating the hierarchical relationships among databases, tables, and columns. This representation includes crucial details such as data types, primary keys, and even example values, providing a richer context for the LLM compared to traditional schema representations.

Experimental results have demonstrated that M-Schema outperforms traditional schema representations like DDL Schema and MAC-SQL Schema, leading to better generalizability and improved accuracy in SQL generation tasks. Studies have shown an average performance improvement of 2.03% across multiple models when using M-Schema. The XiYan-SQL framework, incorporating M-Schema, has achieved state-of-the-art execution accuracy on several benchmark datasets, including 89.65% on the Spider test set, 69.86% on SQL-Eval, 41.20% on NL2GQL, and a competitive 72.23% on the Bird development benchmark.[4] The implementation and examples of M-Schema are available in the GitHub repository associated with the XiYan-SQL project. The success of M-Schema underscores the significance of how database schema is presented to LLMs and suggests that more structured and semantically rich representations can lead to substantial gains in Text-to-SQL accuracy, particularly for complex databases.

### 4.2. Dynamic Few-Shot Learning Strategies

Dynamic Few-Shot Example Retrieval involves fetching and providing the LLM with the most relevant examples from a training set based on the semantic similarity of the input question. Techniques like Jaccard Similarity or cosine similarity on embeddings can be used to measure the similarity between the input question and the questions in the example set. In some cases, synthetic summaries of SQL queries are used instead of the raw SQL to improve the matching process.[6]

### 4.3. BASE-SQL Implementation and Performance[7]

BASE-SQL is a pipeline-based method for Text-to-SQL conversion that uses fine-tuned open-source Large Language Models (LLMs), such as Qwen2.5-Coder.  This approach has four main components that work sequentially.

Schema Linking: The first component focuses on identifying and filtering the database tables relevant to the user's natural language query.  Instead of complex column linking, BASE-SQL uses a fine-tuned model to determine the necessary tables, aiming to reduce errors and improve SQL accuracy.

Candidate SQL Generation: The second component generates potential SQL statements using fine-tuned open-source models. A key feature here is the intentional introduction of "noise" (e.g., adding redundant tables) during generation to simulate real-world complexities and make the model more robust. This often involves Low-Rank Adaptation (LoRA) for efficient fine-tuning with fewer samples.

SQL Revision: The third component corrects logic or syntax errors in the generated SQL. It does this by evaluating the candidate SQLs against their execution results and by considering two schema representations: one from the schema linking step and a full schema representation.

SQL Merge Revision: The final component is used when there are inconsistencies in the execution results of the revised SQL queries. In these cases, the method merges the results to produce a more accurate final SQL statement, rather than just picking one of the candidates.

Experimental results show that BASE-SQL achieves good accuracy, with 67.47% on the BIRD development set and 88.9% on the Spider test set. These results are a notable improvement over other methods using open-source models and even outperform some approaches using the closed-source GPT-4o. BASE-SQL is also efficient, requiring an average of only five language model calls per SQL generation. It's considered easy to implement and has a low reproduction cost, as it doesn't need extra data for pre-training. These qualities make BASE-SQL a cost-effective and efficient baseline method for Text-to-SQL, demonstrating the potential of fine-tuned open-source LLMs in this area.

### 4.4. RESDSQL for Complex Schemas[9]

RESDSQL (Ranking-enhanced Encoding plus a Skeleton-aware Decoding framework for Text-to-SQL) is designed to handle the complexities of Text-to-SQL, especially with intricate database schemas. The main idea is to decouple the two core challenges in Text-to-SQL: schema linking (identifying relevant tables and columns) and skeleton parsing (understanding the SQL query's structure and keywords).

To achieve this decoupling, RESDSQL uses a ranking-enhanced encoder and a skeleton-aware decoder.

The encoder is enhanced by injecting only the most relevant schema items into the sequence-to-sequence model, instead of the entire, often unordered, schema. This pre-selection of relevant schema items, done through a separate schema linking process, reduces the load on the main model during SQL parsing. To get this ranked sequence, RESDSQL uses a cross-encoder trained to classify tables and columns based on the natural language question.  The classifier's probabilities are used to rank and filter schema items, creating a focused schema sequence for the main encoder.

The decoder is skeleton-aware, meaning it first generates the SQL query's skeleton (keywords and structure) before generating the full SQL with table and column names. This two-step decoding takes advantage of the fact that skeleton parsing is generally easier than full SQL parsing, and the generated skeleton guides the subsequent SQL parsing.

RESDSQL has shown promising performance and robustness on the Spider benchmark and its variants, which test the parser's resilience to changes in database schema or natural language questions.  The framework also includes schema matching techniques like substring matching, where cell values from the database are compared to words in the question, with heuristics to refine these matches.  By decoupling schema linking and skeleton parsing, RESDSQL aims to simplify the Text-to-SQL task, particularly with many schema items and complex logic, making it suitable for complex database schemas. The source code for the implementation of this framework using PyTorch is also available.[8]

## 5. Addressing Current Limitations and Future Research Directions

While the research documented and the further investigations conducted for this report highlight significant progress in Text-to-SQL conversion, several limitations persist, particularly concerning the exclusive use of GPT-3.5-turbo-16k in the initial experiments and the challenges posed by large, complex real-world databases. These limitations point towards several promising avenues for future research.

### 5.1. Limitations Encountered with GPT-3.5 and Potential of Other LLMs

The initial research was primarily conducted using the GPT-3.5-turbo-16k model, which, while powerful, limited the ability to perform a comprehensive comparative analysis with other advanced models such as GPT-4 and LLaMA. Subsequent research has indicated that GPT-4 often demonstrates superior accuracy in Text-to-SQL tasks compared to GPT-3.5.[1] The rapid advancements in LLM technology and the increasing availability of powerful open-source models present significant opportunities for advancing the field.

### 5.2. Challenges with Large and Complex Real-World Databases

The complexity of schemas in large industrial databases presented significant accuracy challenges in the initial research, underscoring the need for adaptive schema handling methods. This observation is consistent with findings in other studies, which highlight the limitations of current Text-to-SQL systems when dealing with the scale and intricacy of real-world databases.[1] We can see in the original paper that even the highlighted strategies work and could be compared against each other in the given Mondial dataset. When it came to an actual industrial database (IndDB), the accuracy was near zero for all except using the DANKE keyword search. A key challenge is the inability of current methods to incorporate all relevant table structure information within a single prompt due to the sheer size of the database schema.[11]

### 5.3. Recommended Future Research Avenues

Based on the findings of this research and the broader trends in the field, several future research avenues are recommended:

- Conducting deeper investigations into the integration of RESDSQL to leverage its advanced schema matching and decoupling techniques for handling the complexities of real-world database schemas.[8]
- Developing and evaluating novel adaptive schema handling methods specifically designed to address the challenges posed by large and intricate real-world databases, potentially drawing inspiration from the mschema representation.[5]
- Further exploring the incorporation of Few-Shot Similar Pairs with more sophisticated retrieval mechanisms and larger, more diverse training datasets to solidify its observed potential for enhancing accuracy.[6]
- Investigating the feasibility and benefits of implementing a keyword-search method that stores table and column names along with their possible natural language references to improve the precision of schema linking.
- Undertaking a detailed study of the BASE-SQL pipeline, potentially replicating its implementation and evaluating its performance across various datasets, including more complex industrial databases like IndDB.[7]

## 6. Conclusion

In conclusion, this Directed Studies project has explored various strategies for Text-to-SQL conversion, replicating key findings from recent research and investigating potential enhancements. The initial evaluation of LangChain, C3, and DIN-based frameworks on the Mondial dataset provided a baseline for understanding the strengths and weaknesses of existing approaches. Experiments with M-Schema and Few-Shot Learning demonstrated the potential for improved accuracy, particularly in handling complex queries, although challenges remain in consistently achieving significant gains. The C3+ framework, incorporating M-Schema and an SQL Validator, showed promise but requires further refinement to fully realize its potential.

The research also highlighted the limitations of current Text-to-SQL systems, especially when dealing with the complexities of real-world databases and the constraints of specific LLMs. To address these limitations, future research should explore advanced techniques such as RESDSQL, novel adaptive schema handling methods, and more sophisticated Few-Shot Learning strategies. Additionally, investigations into keyword-search methods and detailed studies of the BASE-SQL pipeline could yield valuable insights. By pursuing these research directions, the field of Text-to-SQL can continue to advance, leading to more effective and user-friendly database interaction tools.

## 7. References

### Works cited

1. Text-to-SQL Meets the Real-World, https://www.scitepress.org/Papers/2024/125552/125552.pdf
2. XiYan-SQL: A Multi-Generator Ensemble Framework for Text-to-SQL - arXiv,, https://arxiv.org/html/2411.08599v1
3. XiYan-SQL Revolutionizes NL2SQL Tasks with State-of-the-Art Framework - AZoAi, https://www.azoai.com/news/20241120/XiYan-SQL-Revolutionizes-NL2SQL-Tasks-with-State-of-the-Art-Framework.aspx
4. XGenerationLab/XiYan-SQL: A MULTI-GENERATOR ENSEMBLE FRAMEWORK FOR NATURAL LANGUAGE TO SQL - GitHub, accessed April 14, 2025, https://github.com/XGenerationLab/XiYan-SQL
5. XGenerationLab/M-Schema: a semi-structure representation of database schema - GitHub, accessed April 14, 2025, https://github.com/XGenerationLab/M-Schema
6. Enhancing Text-to-SQL With Synthetic Summaries: A Few-Shot Learning Approach, accessed April 14, 2025, https://www.timescale.com/blog/enhancing-text-to-sql-with-synthetic-summaries
7. BASE-SQL: A powerful open source Text-To-SQL baseline approach - arXiv, accessed April 14, 2025, https://arxiv.org/html/2502.10739v1
8. The Pytorch implementation of RESDSQL (AAAI 2023). - GitHub, https://github.com/RUCKBReasoning/RESDSQL
9. RESDSQL: Decoupling Schema Linking and Skeleton Parsing for Text-to-SQL | Proceedings of the AAAI Conference on Artificial Intelligence, https://ojs.aaai.org/index.php/AAAI/article/view/26535
10. eosphoros-ai/Awesome-Text2SQL - GitHub, accessed April 14, 2025, https://github.com/eosphoros-ai/Awesome-Text2SQL
11. Large Language Model Enhanced Text-to-SQL Generation: A Survey, arxiv.org, https://arxiv.org/html/2410.06011v1#:~:text=Due%20to%20the%20sheer%20size,to%20generate%20correct%20SQL%20queries.