# SQL-of-Thought: Multi-agentic Text-to-SQL with Guided Error Correction

**Saumya Chaturvedi**
Max Planck Institute for Software Systems
Saarbrücken, Germany
schaturv@mpi-sws.org

**Aman Chadha**
AWS GenAI
Santa Clara, CA, USA
hi@aman.ai

**Laurent Bindschaedler**
Max Planck Institute for Software Systems
Saarbrücken, Germany
bindsch@mpi-sws.org

## Abstract

Converting natural language queries into SQL queries is a crucial challenge in both industry and academia, aiming to increase access to databases and large-scale applications. This work examines how in-context learning and chain-of-thought can be utilized to develop a robust solution for text-to-SQL systems. We propose SQL-of-Thought: a multi-agent framework that decomposes the Text2SQL task into schema linking, subproblem identification, query plan generation, SQL generation, and a guided correction loop. Unlike prior systems that rely only on execution-based static correction, we introduce taxonomy-guided dynamic error modification informed by in-context learning. SQL-of-Thought achieves 91.59% Execution Accuracy on the Spider dataset and similar state-of-the-art results on its variants, combining guided error taxonomy with reasoning-based query planning.[1]

## 1 Introduction

Text-to-SQL (NL2SQL) has emerged as a crucial problem in both research and real-world applications, enabling non-technical users to query structured databases through natural language. While early sequence-to-sequence and schema-aware models improved accessibility, they often struggled with generalization, resulting in subpar performance. Recent advances in large language models (LLMs) and prompting techniques have significantly raised performance, with methods such as DIN-SQL [11] and DAIL-SQL [5] decomposing the task into subtasks. However, as highlighted by Li et al. [7] and Biswal et al. [1], even the most advanced systems remain brittle when dealing with realistic queries. This is because execution-based feedback alone cannot correct logically incorrect, yet syntactically valid, SQL queries.

Multi-agent approaches [21, 12, 14] and reasoning-guided prompting [19, 10] have shown potential to bridge this gap. Multi-agent systems enhance modularity and specialization; however, their error correction still relies heavily on execution signals or static re-generation. On the other hand, reasoning-based methods improve query planning, yet unguided reasoning can often introduce new errors or inefficiencies. What is needed is a systematic way to combine multi-agent decomposition with structured reasoning and interpretable correction.

We tackle this issue using SQL-of-Thought, a multi-agent framework designed to enhance SQL queries. This framework incorporates several key components: (i) specialized agents for schema

[1] Github Code Repository

Preprint. Under review.

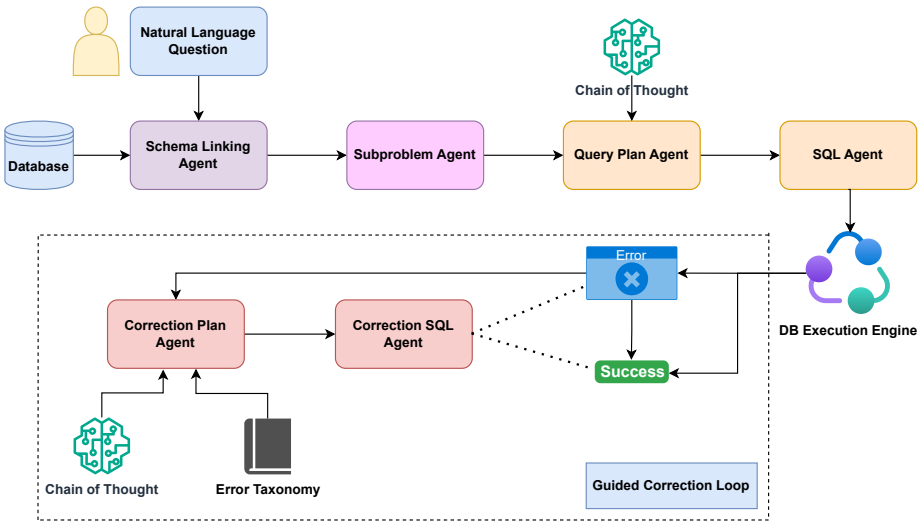


Figure 1: Architecture of SQL-of-Thought. The process begins as the user asks a natural language question. The relevant schema is fetched from the database, and sent alongwith the question to Schema Linking agent, which extracts the useful tables and columns needed to solve the question. This cropped schema and the question is sent to the Subproblem agent, which divides the problem into smaller, clause-wise subproblems. Then the Query Plan Agent is invoked, using Chain of Thought to procedurally generate an executable Query Plan. This query plan is used by the SQL Agent to generate a valid SQL solution, which is then executed on the DB Execution engine. If the execution fails, the correction loop is invoked, with the execution error, error taxonomy, and incorrect SQL sent to the Correction Plan Agent, to investigate the error and generate a Correction Plan using a CoT-guided rectification mechanism to fix it. The Correction SQL Agent uses this plan to generate the correct SQL query. The Guided Correction Loop is invoked until the query is successful or maximum attempts are finished.

linking, identifying subproblems, generating query plans, synthesizing SQL, and correcting errors; (ii) a taxonomy-guided error correction loop that categorizes SQL failure modes and rectifies them through a Chain of Thought reasoning process; and (iii) an analysis comparing reasoning and non-reasoning models across different agent roles. Our framework achieves state-of-the-art execution accuracy on Spider [22], Spider-Realistic [3], and Spider-SYN [4], while also showcasing effective cost-performance trade-offs through the use of hybrid models. These results demonstrate that compact, guided reasoning, paired with structured error feedback, is a more reliable approach for SQL querying than solely relying on execution-based refinement.

We make the following contributions:

- SQL-of-Thought: A novel multi-agent solution for NL2SQL, complete with Schema Linking, Subproblem identification, Chain of Thought (CoT) for query plan generation, SQL generation, and an explicit error correction loop.
- Guided Error Modification driven by a comprehensive Error Taxonomy and In-Context Learning
- A detailed analysis of Reasoning and Non-reasoning model choices for SQL-of-Thought
- State of the Art results on Spider [22] benchmark and its variations: Spider-Realistic [3], and Spider-SYN [4].

## 2 Related Work

### 2.1 History of NL2SQL

Initial works for Text-to-SQL used major sequence-to-sequence architectures such as PLMs (Programming Language Models), for example RESDSQL [8], which identified relevant schema from the question and then used it to construct the SQL query. Other choices were GNNs (Graph Neural Networks), Recurrent Neural Networks (RNNs), Transformers, and most recently LLMs for natural language to SQL conversion. With the advent of LLMs, prompt engineering on powerful models like GPT [2, 9] and Claude enabled fine-grained and faster solutions for NL2SQL. Shi et al [16] highlights how LLM-based solutions outperformed all other alternatives for programming text-to-SQL.

DIN-SQL [11] attempted in-context learning with error correction, decomposing subproblems, and defining different prompt templates to tackle each subtask. However, their method only regenerates prompts in case of failure, without specific error information. DAIL-SQL [5] designs sophisticated SQL-code style prompt templates to encode the question and database schema. These templates provide few-shot examples for schema linking and SQL query generation, utilizing GPT-4 to generate solutions. DAILSQL-SC employs self-consistency to postprocess solutions for further refining.

Some studies [7] discuss production readiness and evaluation frameworks for NL2SQL using various prompting methods, again verifying that LLM-based approaches generally outperform pretrained task-specific models through higher reliability and generalization. They classify queries into four categories: subqueries, joins, logical connectors, and order-by clauses, to measure model efficacy across increasing levels of difficulty. The TAG framework [1] showed how existing models fail on complex real-world queries, highlighting that most NL2SQL models and benchmarks are only effective for roughly 20% of realistic user queries. They introduce a reasoning framework that decomposes the problem into three stages: (i) query synthesis through NL2SQL parsing, (ii) query execution over the underlying database, and (iii) answer generation, where an LLM integrates both the natural language query and execution results to produce a final human-readable answer.

The TAG framework [1] demonstrated how existing models struggle with complex real-world queries and proposed steps for query synthesis, execution, and answer generation. Multi-agent frameworks, such as Wang et al. [20], introduced syntax validation and execution optimization, but relied mainly on execution feedback.

### 2.2 Agentic Approaches

Tool-SQL [21] pioneered the tool-first agentic framework, where an LLM-based agent iteratively generates and refines SQL queries with the aid of two tools: a database retriever to resolve condition mismatches and an error detector for stricter constraint validation. The approach primarily addressed database mismatches, such as missing highlights, altered paraphrasing, and similar issues.

Chase SQL [12] defines an elaborate framework consisting of value retrieval, multiple candidate generators and query fixers, and a selection agent. The candidates generate a query plan by prompting the LLM through an "EXPLAIN" keyword. The multi-agent framework proposed by Shao et al. [14] introduces a four-agent system where specialized roles (developer, researcher, executor, specialist) collaborate on simple and complex NL2SQL queries. By combining example retrieval, chain-of-thought reasoning, syntax validation, and error-driven refinement, the framework improves both correctness and execution efficiency.

### 2.3 Reasoning guided solutions to NL2SQL

ACT-SQL [23] experiments with prompting styles and few-shot exemplar selection strategies to automate prompt selection for natural language to SQL solutions, making the process cheap and time-saving. Think2SQL [10] and related methods explored reasoning for text-to-SQL by combining zero-shot prompting, supervised finetuning, and reinforcement learning. They asked whether reasoning improves text-to-SQL, how best to train reasoning-capable models, and whether execution-based rewards are sufficient. Their results show mixed evidence: reasoning models often perform better on query planning but not consistently across datasets. They also simplified schema linking by manually providing subsets, limiting the scope of evaluation. Tai et al. [19] highlight Chain-of-

Thought prompting techniques, which achieve significant gains on the Spider dataset, and mention how unnecessarily detailed reasoning can propagate errors.

Our work differs in that we employ a chain of thought reasoning approach, not only for planning but also for schema linking, subproblem division and error correction, guided by an explicit taxonomy. We also evaluate our approach on a variety of models, both reasoning and general-purpose LLMs, to verify the importance of model CoT strength.

## 3 Methodology

The proposed SQL-of-Thought framework is implemented as an LLM-driven text-to-SQL pipeline that generates an executable SQL query $Y$. The process can be formalized as:

$$Y = LLM(Q, S, C, P, T|\theta)$$

where $Q$ is the natural language question, $S$ is the linked schema, $C$ denotes clause-specific subproblems, $P$ is the query plan and $T$ represents the error taxonomy used for CoT-informed error correction. $LLM(.|\theta)$ is a language model parameterized by $\theta$.

### 3.1 SQL-of-Thought

A detailed composition of the SQL-of-Thought framework is as follows:

**Schema Linking Agent** The Schema Linking Agent parses the natural language question in conjunction with the database schema (identified via *db_id*) to identify the relevant tables and columns required for answering the query. In addition, it extracts structural information such as primary keys, foreign keys, and join relationships. This representation forms the foundation for subsequent steps by constraining SQL generation to schema-relevant entities.

**Subproblem Agent**. Given the natural language question and the schema-linked output, the Subproblem Agent decomposes the query into clause-level subproblems (e.g., WHERE, GROUP BY, JOIN, DISTINCT, ORDER BY, HAVING, EXCEPT, LIMIT, UNION). Each identified clause is expressed as a key–value pair in a structured JSON object, where the key is the clause type and the value is the partially completed clause expression. This decomposition provides a modular representation of the query intent, enabling downstream agents to reason over smaller, well-defined units.

**Query Plan Agent**. The Query Plan Agent generates a step-by-step execution plan that maps the user's intent to the schema and subproblems. Unlike prior work such as Chase-SQL [12], which treats planning as a surface-level mapping, our agent is explicitly prompted to perform chain-of-thought reasoning to explain intermediate decisions. This structured reasoning encourages deeper alignment between the question, subproblems, and the final SQL query. The Query Plan Agent produces only the procedural plan and is explicitly restricted from generating executable SQL at this stage.

**SQL Agent**. The SQL Agent consumes the natural language question and the query plan to generate the executable SQL query. Post-processing removes extraneous artifacts such as trailing semicolons or natural language fragments, ensuring the query is syntactically valid. The generated query is then executed against the database, and the result is compared with the ground-truth answer. If the query fails to match, the pipeline transitions to the correction loop.

**Correction Plan Agent**: The Correction Plan Agent initiates the correction loop by analyzing the failed SQL query in the context of the natural language question, schema, and execution results. Unlike systems such as DIN-SQL or DAIL-SQL [5, 11], which rely solely on execution feedback, this agent is additionally guided by an error taxonomy derived from Shen et al. [15]. The taxonomy categorizes common error modes (e.g., schema mismatches, join inconsistencies, aggregation misuse), and the agent produces a chain-of-thought plan describing how to resolve the identified errors. Inspired by reflexive learning approaches [17], the agent iteratively generates structured feedback that informs the correction SQL generation process.

**Correction SQL Agent**: The Correction SQL Agent takes as input the correction plan, the question, the schema, and the incorrect SQL query. Based on the structured guidance, it regenerates the SQL query while avoiding the previous errors. The corrected query is re-executed on the database, and if the result still does not match, the correction loop is re-entered until convergence or until the maximum number of correction attempts is reached.

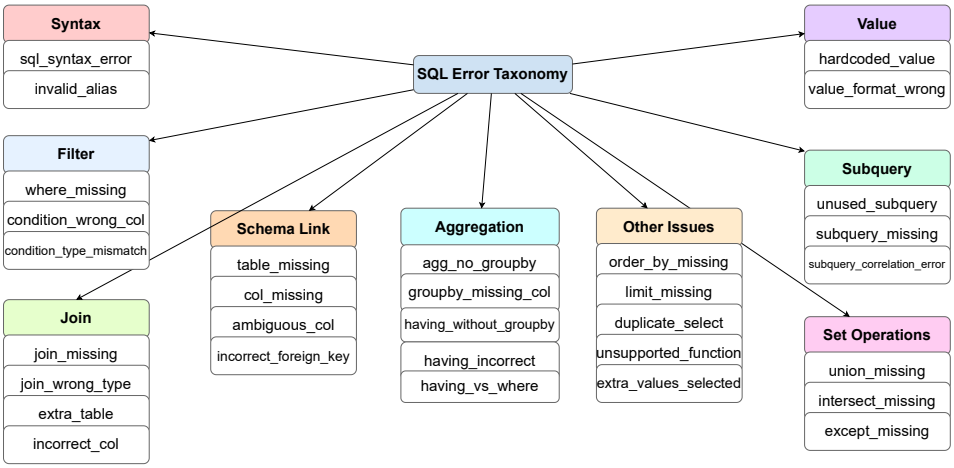


Figure 2: Error Taxonomy proposed for SQL-of-Thought. This taxonomy has 9 categories and 31 sub-categories of logical errors to be identified and rectified by LLMs.

### 3.2 Error Taxonomy

The error taxonomy adopted in our correction loop, shown in Figure 2, is derived from the classification proposed in Shen et al. [15], which systematically categorizes standard failure modes encountered in NL2SQL systems. Our proposed taxonomy expands on this by creating systematic categories of errors that can be identified and corrected by LLMs, as well as comprehensive subtypes for each. We provide concise error codes instead of lengthy explanations to facilitate easier identification and prevent overflowing the LLM context window. Our taxonomy spans a broad range of issues, including **syntax errors** (e.g., invalid aliases or malformed SQL), **schema linking errors** (e.g., missing or ambiguous columns, incorrect foreign keys), and **join-related mistakes** (such as missing joins, wrong join types, or inclusion of extra tables). It further covers errors in **filter** conditions (e.g., incorrect columns in the WHERE clause, type mismatches), **aggregation logic** (e.g., missing GROUP BY, misuse of HAVING), as well as **value representation errors** like hard-coded values or format mismatches. More advanced categories capture failures in **subquery** formulation (e.g., unused or incorrectly correlated subqueries), **set operations** (union, intersection, except), and other structural oversights, such as missing ORDER BY or LIMIT clauses. By explicitly codifying these error types, the taxonomy provides a fine-grained diagnostic lens that enables the correction plan agent to reason not just about what failed, but why it failed within the SQL generation pipeline, ranging from basic syntax issues to complex sub-query and join formulation errors, providing specific guidance for correction.

Incorporating this taxonomy into the correction loop enables the system to move beyond coarse execution-based feedback (as used in DIN-SQL [11] and DAIL-SQL [5]) and instead provide interpretable, linguistically grounded guidance for iterative refinement. We prompt the correction plan agent with the summarized error taxonomy, along with a chain-of-thought reasoning template, which equips the agent to identify root causes of failure, align errors with schema or syntax constraints, and propose concrete repair strategies. This approach aligns with findings in reflexive learning approaches [17], where LLMs benefit from verbalized self-feedback in enhancing decision-making for programming tasks. Thus, by leveraging the structured taxonomy from Shen et al. [15], the multi-agent system establishes a principled loop of error detection, diagnosis, and guided correction, thereby closing the gap between raw execution feedback and systematic, explainable query refinement.

### 3.3 Experimental Setup

#### 3.3.1 Dataset

We utilize the Spider dataset [22] for evaluating our multi-agent framework. Spider offers a diverse range of database schemas and tasks, including 20 database settings and 1034 text-SQL pairs in its dev split. We avoid the BIRD-SQL benchmark due to the high quantity of annotation errors, as reported in Shen et al [15]. BIRD-SQL has fewer set operation-based questions than SPIDER, and also presents an evidence field for queries, which can add to hallucination while query generation. For these purposes, we test on Spider and its popular variant, Spider Realistic [3], which is an evaluation set based on the Spider dev set with explicit mentions of column names removed, containing 508 samples. Spider SYN [4] is a challenging variant of the Spider evaluation dataset. Spider- SYN is constructed by manually modifying natural language questions with synonym substitutions.

#### 3.3.2 Evaluation Metrics

While we observe metrics such as Exact Match and Valid SQL Generation during our experiments, we only consider Execution Accuracy for holistic evaluation. One natural language question could have multiple SQL queries that answer it. LLMs frequently oversimplify, and in the case of Text-to-SQL tasks, they were found to assign variables to results from subqueries, which is why an Exact Match (EM) is not an accurate representation of the accuracy of the process. Thus, we adopt Execution Accuracy (EA) as the metric to evaluate our process. EA is a boolean field that compares the execution results of the generated query with the execution results of the Gold SQL label that exists in the dataset. Since the EA calculation requires the generated SQL query to be executed against Spider's SQLite files, we also adopt SQLite [18] for query generation.

#### 3.3.3 Hardware & Configuration

We perform all experiments on a machine equipped with two NVIDIA H100 GPUs, each with 80 GB of HBM, and configured with Debian Ubuntu, running PyTorch version 2.5.1.

### 3.4 Guided Error modification

Execution-based feedback alone is insufficient for error correction in NL2SQL. In our experiments, we observed that when using models such as Claude 3 Opus or GPT-4o-mini within the SQL-of-Thought framework, 95–99% of generated queries were already syntactically valid. Thus, traditional database execution errors were rare, and the predominant failures stemmed instead from *intent mismatches*, logically incorrect but syntactically valid queries, or errors involving incorrect or extraneous values through various join and limit clauses. Providing only execution traces [11, 5] therefore offers little guidance for effective correction. To address this, we integrate the error taxonomy defined in Figure 2, which enumerates a wide range of SQL error types. By exposing the model to these structured error categories, the correction loop enables the agent to identify better and rectify subtle logical and semantic flaws. Empirically, on our strongest configuration (Claude 3 Opus + SQL-of-Thought), omitting the correction loop led to a 10% increase in incorrect queries compared to the whole system with guided error modification, as demonstrated through ablation on a 100-sample evaluation set in Table **??**.

### 3.5 Query Plan and then SQL Generation

We also find that reasoning-driven LLMs perform significantly better when first producing a structured query plan before attempting SQL generation. The query plan serves as an intermediate reasoning step, allowing the model to explicitly organize relevant schema elements, logical conditions, aggregations, and subproblems before constructing the final query. This structured decomposition reduces hallucinations and improves alignment between the natural language intent and the SQL output. In our ablation study with Claude 3 Opus + SQL-of-Thought, bypassing the query plan agent and directing the SQL agent to generate queries directly increased the rate of incorrect queries by 5% on a 100-sample test set, shown in Table **??**. These results highlight the importance of staged reasoning—query plan synthesis followed by SQL generation as a critical design choice for improving correctness in NL2SQL systems.

## 4 Results

We present the results of the proposed SQL-of-Thought framework on the Spider [22] and Spider Realistic [3] benchmarks (wherever results were available) in Table 1, highlighting the execution accuracy reported by earlier works and demonstrating how SQL-of-Thought outperforms existing methods. We also report an Execution Accuracy of **82.01%** on the Spider SYN [4] benchmark. For all our experiments, we set the temperature to 0, and set top_p and other hyperparameters to default. These results are based on the Claude Opus 3 model, which serves as the foundation for evaluating our SQL-of-Thought framework. SQL-of-Thought reports an Execution Accuracy of **91.59%** on Spider [22] Benchmark and **90.16%** Execution Accuracy on Spider Realistic [3] benchmark. We also wish to report that for all our experiments, we observe a Valid SQL generation rate of 94%-99% with SQL-of-Thought.

| **Method** | **Spider** | **Spider-Realistic** |
|---|---|---|
| ChatGPT [2] | 74.4 | - |
| GPT-4 (OpenAI 2023) [9] | 72.3 | - |
| ACT-SQL + ChatGPT (Zhang et al. 2023) [23] | 80.4 | 75.8 |
| ACT-SQL + GPT-4 (Zhang et al. 2023) [23] | 82.9 | - |
| DIN-SQL + GPT-4 (Pourreza and Rafiei 2024) [11] | 82.8 | 78.1 |
| DAIL-SQL + GPT-4 (Gao et al. 2023) [5] | 83.1 | 75.6 |
| DAIL-SQL + GPT-4 + SC (Gao et al. 2023) [5] | 83.6 | 75.2 |
| MAC-SQL + GPT-4 (Wang et al. 2024) [20] | 86.8 | - |
| Tool-SQL + GPT-4 [21] | 86.9 | 82.9 |
| Chase SQL [12] | 87.6 | - |
| **SQL-of-Thought (with Claude Opus 3)** | **91.59** | **90.16** |

Table 1: Execution Accuracy of prior methods and our proposed SQL-of-Thought on Spider (Dev split) and Spider-Realistic (Dev split). Dashes indicate that the corresponding result was not reported by the authors.

Table 4 denotes the results obtained when different models were employed as base LLMs in SQL-of-Thought on a 100-sample mini-batch from the Spider Dev dataset. We also evaluated the framework once with just the initial sequential flow, removing the self-correction loop, and once with direct SQL generation after Schema Linking and Subproblem Agents, removing the CoT Query Plan generation during both the initial and correction flows. The accuracy falls at least 5% without a Chain-of-Thought Query Plan before the SQL generation, and at least 8-10% for all models when the correction loop is not invoked. The models are listed in decreasing order of accuracy, noting that Claude Opus 3 performs the best, with GPT-5, GPT-4o-mini [9], and finally GPT-3.5 [2] producing **89%, 87%** and **67%** accuracy, respectively.

We also report that a single run on Spider Dataset for SQL-of-Thought with Claude Opus 3, took 5 hours to run, and $42.58 token cost at $15/MTok (million tokens).

| **Model** | SQL-of-Thought | W/o Error Correction | W/o Query Plan Generation |
|---|---|---|---|
| **Claude 3 Opus** | **95** | **85** | **90** |
| GPT 5 | 89 | 85 | 88 |
| GPT 4o Mini | 87 | 72 | 79 |
| GPT 3.5 | 67 | 59 | 73 |

Table 2: Execution Accuracy Results for the Multiagent SQL model on the first 100 samples of Spider Dataset, evaluated using various base models and removal of correction loop, and query plan generation step to highlight the results.

# 5 Discussions

## 5.1 Reasoning vs. Non-Reasoning Models

Reasoning models demonstrated clear advantages in nearly all specialized agentic tasks, including schema linking, query planning, chain of thought, and clause identification. Non-reasoning models often failed to decompose queries correctly and produced valid but logically wrong SQL. Errors such as confusing ColumnA from TableA and listing it under ColumnB, forgetting to include aggregate groupby clauses, selecting extraneous or fewer columns, and more were prevalent. However, reasoning alone did not guarantee improvements in all cases. For example, unguided reasoning without an error taxonomy led to worse correction and often resulted in the repetition of the same correction steps in different attempts. This acutely highlights how the dynamic guided correction loop complemented the Chain-of-Thought style prompting and helped the model learn from direct examples and an error playbook.

## 5.2 Learnings

Using Claude Opus 3 instead of GPT improved accuracy. The guided correction loop, where the agent relied on an error taxonomy, reduced repetition and improved logical corrections. Reasoning-based query plan generation before SQL synthesis also consistently improved results, as LLMs have been shown to be better at formulating reasoning for mathematical, programming, and planning tasks. In our ablation study with Claude 3 Opus + SQL-of-Thought, bypassing the query plan agent and directing the SQL agent to generate queries directly increased the rate of incorrect queries by 5% on a 100-sample test set, shown in Table 4. These results highlight the importance of staged reasoning: query plan synthesis followed by SQL generation as a critical design choice for improving correctness in NL2SQL systems.

## 5.3 Lessons from Unsuccessful Ablations

Not all design choices improved accuracy; several variants underperformed, highlighting the importance of guided reasoning, as mentioned below:

- A critic loop that applied the full error taxonomy in a free-form way and sent errors directly to the SQL agent underperformed. Accuracy improved when errors were first routed through a query plan agent, showing that LLMs struggle with unguided debugging and benefit from at least one structured reasoning step.
- Increasing temperature above 0 for GPT-4o added surface diversity but reduced plan faithfulness, producing more invalid joins and clause misuse.
- Adding specific rules in the agent prompts for specific clauses (e.g., JOIN, LIMIT) inflated the context window and often distracted the model with irrelevant details, lowering accuracy.
- An ablation design with multiple repair agents, each addressing a specific error type of a subproblem and feeding into an aggregation agent to generate the final SQL solution, failed. Independent edits often conflicted, and the merge process produced incoherent SQL.
- Carrying history across correction attempts through a shared scratchpad expanded the context window, increased latency and API cost, and amplified repetition and schema drift, leading to lower accuracy.

## 5.4 Token Cost and other Metrics

LLMs are better and faster at NL2SQL [16, 7], but the API costs are very expensive. Especially in a multi-agent framework or any multiple-pipeline system, the cost for each sample is a lot. Increasing accuracy comes at the cost of compute and expensive processes. A single run on the Spider benchmark takes about 42.58$ for Claude Opus 3, and 44.2$ for GPT models. Switching to cheaper models led to a dip in accuracy, but good-performing models were more expensive. We tried to find a middle ground that is both cost-effective and high-performing. With various combinations of Claude and GPT model inference for different agents, we discovered that tasks such as Schema Linking, Query Plan Generation, and Correction Plan Generation required higher reasoning power and failed when using non-reasoning models. This meant that the Subproblem, SQL, and Correction SQL Agents

could use non-reasoning models and still perform at par. This approach enabled a hybrid model of Claude Opus for reasoning-intensive agents and GPT-4o for the other agents to create a good middle ground at approximately a $30 cost for an entire dataset run, achieving 85% Execution Accuracy (for a 100-sample ablation).

Additionally, we also tested SQL-of-Thought on open-source models, such as LLama-3.1-8B-Instruct [6] and Qwen2.5-1.5B [13], and similar variations. We found that not only do these models exhibit high latency, taking up to three times longer for evaluation, but they also underperform significantly, achieving only about 45.3% accuracy on the 100-sample set. These models had trouble generating SQL (led to long strings of repeated hallucination), missing column names across different tables, and were overall not suited for use in an NL2SQL framework. Future work can improve the cost and efficacy of such a framework by fine-tuning SLMs (small language models) on individual agentic tasks. For example, an error database can be designed to fine-tune the Correction Loop agents. Spider [22] provides detailed fields for each query, including whether it uses clauses such as LIMIT, INTERSECT, GROUPBY, etc. These clauses can be used to create datasets for fine-tuning the Subproblem Agent.

## 6 Limitations

Our evaluation is limited to the Spider [22] benchmark and its variants, which may not fully capture the challenges of real-world databases. The error taxonomy, although effective, has not been exhaustively validated across diverse query structures. The multi-agent design also increases inference costs, relying heavily on closed-source reasoning LLMs. Future work can improve both cost and efficacy by fine-tuning small language models (SLMs) for individual agentic tasks, such as building an error database to fine-tune correction loop agents or utilizing Spider's clause-level annotations (e.g., LIMIT, INTERSECT, GROUPBY) to fine-tune the Subproblem Agent.

## 7 Conclusion

We introduced SQL-of-Thought, a multi-agent NL2SQL framework that combines schema linking, subproblem identification, chain-of-thought query planning, SQL generation, and taxonomy-guided error correction. The framework achieves state-of-the-art execution accuracy on Spider benchmarks, showing that structured reasoning and guided correction are more effective than execution-only feedback. Future work should extend evaluation to real-world datasets and explore fine-tuned SLMs for cost-efficient deployment.

## References


[1] Asim Biswal, Liana Patel, Siddarth Jha, Amog Kamsetty, Shu Liu, Joseph E. Gonzalez, Carlos Guestrin, and Matei Zaharia. Text2sql is not enough: Unifying ai and databases with tag, 2024. URL `https://arxiv.org/abs/2408.14717`.

[2] Tom B. Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel M. Ziegler, Jeffrey Wu, Clemens Winter, Christopher Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners, 2020. URL `https://arxiv.org/abs/2005.14165`.

[3] Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. Structure-grounded pretraining for text-to-sql. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.naacl-main.105. URL `http://dx.doi.org/10.18653/v1/2021.naacl-main.105`.

[4] Yujian Gan, Xinyun Chen, Qiuping Huang, Matthew Purver, John R. Woodward, Jinxia Xie, and Pengsheng Huang. Towards robustness of text-to-sql models against synonym substitution, 2021. URL `https://arxiv.org/abs/2106.01065`.

[5] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. Text-to-sql empowered by large language models: A benchmark evaluation, 2023. URL https://arxiv.org/abs/2308.15363.

[6] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, Amy Yang, Angela Fan, Anirudh Goyal, Anthony Hartshorn, Aobo Yang, Archi Mitra, Archie Sravankumar, Artem Korenev, Arthur Hinsvark, Arun Rao, Aston Zhang, Aurelien Rodriguez, Austen Gregerson, Ava Spataru, Baptiste Roziere, Bethany Biron, Binh Tang, Bobbie Chern, Charlotte Caucheteux, Chaya Nayak, Chloe Bi, Chris Marra, Chris McConnell, Christian Keller, Christophe Touret, Chunyang Wu, Corinne Wong, Cristian Canton Ferrer, Cyrus Nikolaidis, Damien Allonsius, Daniel Song, Danielle Pintz, Danny Livshits, Danny Wyatt, David Esiobu, Dhruv Choudhary, Dhruv Mahajan, Diego Garcia-Olano, Diego Perino, Dieuwke Hupkes, Egor Lakomkin, Ehab AlBadawy, Elina Lobanova, Emily Dinan, Eric Michael Smith, Filip Radenovic, Francisco Guzmán, Frank Zhang, Gabriel Synnaeve, Gabrielle Lee, Georgia Lewis Anderson, Govind Thattai, Graeme Nail, Gregoire Mialon, Guan Pang, Guillem Cucurell, Hailey Nguyen, Hannah Korevaar, Hu Xu, Hugo Touvron, Iliyan Zarov, Imanol Arrieta Ibarra, Isabel Kloumann, Ishan Misra, Ivan Evtimov, Jack Zhang, Jade Copet, Jaewon Lee, Jan Geffert, Jana Vranes, Jason Park, Jay Mahadeokar, Jeet Shah, Jelmer van der Linde, Jennifer Billock, Jenny Hong, Jenya Lee, Jeremy Fu, Jianfeng Chi, Jianyu Huang, Jiawen Liu, Jie Wang, Jiecao Yu, Joanna Bitton, Joe Spisak, Jongsoo Park, Joseph Rocca, Joshua Johnstun, Joshua Saxe, Junteng Jia, Kalyan Vasuden Alwala, Karthik Prasad, Kartikeya Upasani, Kate Plawiak, Ke Li, Kenneth Heafield, Kevin Stone, Khalid El-Arini, Krithika Iyer, Kshitiz Malik, Kuenley Chiu, Kunal Bhalla, Kushal Lakhotia, Lauren Rantala-Yeary, Laurens van der Maaten, Lawrence Chen, Liang Tan, Liz Jenkins, Louis Martin, Lovish Madaan, Lubo Malo, Lukas Blecher, Lukas Landzaat, Luke de Oliveira, Madeline Muzzi, Mahesh Pasupuleti, Mannat Singh, Manohar Paluri, Marcin Kardas, Maria Tsimpoukelli, Mathew Oldham, Mathieu Rita, Maya Pavlova, Melanie Kambadur, Mike Lewis, Min Si, Mitesh Kumar Singh, Mona Hassan, Naman Goyal, Narjes Torabi, Nikolay Bashlykov, Nikolay Bogoychev, Niladri Chatterji, Ning Zhang, Olivier Duchenne, Onur Çelebi, Patrick Alrassy, Pengchuan Zhang, Pengwei Li, Petar Vasic, Peter Weng, Prajjwal Bhargava, Pratik Dubal, Praveen Krishnan, Punit Singh Koura, Puxin Xu, Qing He, Qingxiao Dong, Ragavan Srinivasan, Raj Ganapathy, Ramon Calderer, Ricardo Silveira Cabral, Robert Stojnic, Roberta Raileanu, Rohan Maheswari, Rohit Girdhar, Rohit Patel, Romain Sauvestre, Ronnie Polidoro, Roshan Sumbaly, Ross Taylor, Ruan Silva, Rui Hou, Rui Wang, Saghar Hosseini, Sahana Chennabasappa, Sanjay Singh, Sean Bell, Seohyun Sonia Kim, Sergey Edunov, Shaoliang Nie, Sharan Narang, Sharath Raparthy, Sheng Shen, Shengye Wan, Shruti Bhosale, Shun Zhang, Simon Vandenhende, Soumya Batra, Spencer Whitman, Sten Sootla, Stephane Collot, Suchin Gururangan, Sydney Borodinsky, Tamar Herman, Tara Fowler, Tarek Sheasha, Thomas Georgiou, Thomas Scialom, Tobias Speckbacher, Todor Mihaylov, Tong Xiao, Ujjwal Karn, Vedanuj Goswami, Vibhor Gupta, Vignesh Ramanathan, Viktor Kerkez, Vincent Gonguet, Virginie Do, Vish Vogeti, Vítor Albiero, Vladan Petrovic, Weiwei Chu, Wenhan Xiong, Wenyin Fu, Whitney Meers, Xavier Martinet, Xiaodong Wang, Xiaofang Wang, Xiaoqing Ellen Tan, Xide Xia, Xinfeng Xie, Xuchao Jia, Xuewei Wang, Yaelle Goldschlag, Yashesh Gaur, Yasmine Babaei, Yi Wen, Yiwen Song, Yuchen Zhang, Yue Li, Yuning Mao, Zacharie Delpierre Coudert, Zheng Yan, Zhengxing Chen, Zoe Papakipos, Aaditya Singh, Aayushi Srivastava, Abha Jain, Adam Kelsey, Adam Shajnfeld, Adithya Gangidi, Adolfo Victoria, Ahuva Goldstand, Ajay Menon, Ajay Sharma, Alex Boesenberg, Alexei Baevski, Allie Feinstein, Amanda Kallet, Amit Sangani, Amos Teo, Anam Yunus, Andrei Lupu, Andres Alvarado, Andrew Caples, Andrew Gu, Andrew Ho, Andrew Poulton, Andrew Ryan, Ankit Ramchandani, Annie Dong, Annie Franco, Anuj Goyal, Aparajita Saraf, Arkabandhu Chowdhury, Ashley Gabriel, Ashwin Bharambe, Assaf Eisenman, Azadeh Yazdan, Beau James, Ben Maurer, Benjamin Leonhardi, Bernie Huang, Beth Loyd, Beto De Paola, Bhargavi Paranjape, Bing Liu, Bo Wu, Boyu Ni, Braden Hancock, Bram Wasti, Brandon Spence, Brani Stojkovic, Brian Gamido, Britt Montalvo, Carl Parker, Carly Burton, Catalina Mejia, Ce Liu, Changhan Wang, Changkyu Kim, Chao Zhou, Chester Hu, Ching-Hsiang Chu, Chris Cai, Chris Tindal, Christoph Feichtenhofer, Cynthia Gao, Damon Civin, Dana Beaty, Daniel Kreymer, Daniel Li, David Adkins, David Xu, Davide Testuggine, Delia David, Devi Parikh, Diana Liskovich, Didem Foss, Dingkang Wang, Duc Le, Dustin Holland, Edward Dowling, Eissa Jamil, Elaine Montgomery, Eleonora Presani, Emily Hahn, Emily Wood, Eric-Tuan Le, Erik Brinkman, Esteban Arcaute, Evan Dunbar, Evan Smothers,

Fei Sun, Felix Kreuk, Feng Tian, Filippos Kokkinos, Firat Ozgenel, Francesco Caggioni, Frank Kanayet, Frank Seide, Gabriela Medina Florez, Gabriella Schwarz, Gada Badeer, Georgia Swee, Gil Halpern, Grant Herman, Grigory Sizov, Guangyi, Zhang, Guna Lakshminarayanan, Hakan Inan, Hamid Shojanazeri, Han Zou, Hannah Wang, Hanwen Zha, Haroun Habeeb, Harrison Rudolph, Helen Suk, Henry Aspegren, Hunter Goldman, Hongyuan Zhan, Ibrahim Damlaj, Igor Molybog, Igor Tufanov, Ilias Leontiadis, Irina-Elena Veliche, Itai Gat, Jake Weissman, James Geboski, James Kohli, Janice Lam, Japhet Asher, Jean-Baptiste Gaya, Jeff Marcus, Jeff Tang, Jennifer Chan, Jenny Zhen, Jeremy Reizenstein, Jeremy Teboul, Jessica Zhong, Jian Jin, Jingyi Yang, Joe Cummings, Jon Carvill, Jon Shepard, Jonathan McPhie, Jonathan Torres, Josh Ginsburg, Junjie Wang, Kai Wu, Kam Hou U, Karan Saxena, Kartikay Khandelwal, Katayoun Zand, Kathy Matosich, Kaushik Veeraraghavan, Kelly Michelena, Keqian Li, Kiran Jagadeesh, Kun Huang, Kunal Chawla, Kyle Huang, Lailin Chen, Lakshya Garg, Lavender A, Leandro Silva, Lee Bell, Lei Zhang, Liangpeng Guo, Licheng Yu, Liron Moshkovich, Luca Wehrstedt, Madian Khabsa, Manav Avalani, Manish Bhatt, Martynas Mankus, Matan Hasson, Matthew Lennie, Matthias Reso, Maxim Groshev, Maxim Naumov, Maya Lathi, Meghan Keneally, Miao Liu, Michael L. Seltzer, Michal Valko, Michelle Restrepo, Mihir Patel, Mik Vyatskov, Mikayel Samvelyan, Mike Clark, Mike Macey, Mike Wang, Miquel Jubert Hermoso, Mo Metanat, Mohammad Rastegari, Munish Bansal, Nandhini Santhanam, Natascha Parks, Natasha White, Navyata Bawa, Nayan Singhal, Nick Egebo, Nicolas Usunier, Nikhil Mehta, Nikolay Pavlovich Laptev, Ning Dong, Norman Cheng, Oleg Chernoguz, Olivia Hart, Omkar Salpekar, Ozlem Kalinli, Parkin Kent, Parth Parekh, Paul Saab, Pavan Balaji, Pedro Rittner, Philip Bontrager, Pierre Roux, Piotr Dollar, Polina Zvyagina, Prashant Ratanchandani, Pritish Yuvraj, Qian Liang, Rachad Alao, Rachel Rodriguez, Rafi Ayub, Raghotham Murthy, Raghu Nayani, Rahul Mitra, Rangaprabhu Parthasarathy, Raymond Li, Rebekkah Hogan, Robin Battey, Rocky Wang, Russ Howes, Ruty Rinott, Sachin Mehta, Sachin Siby, Sai Jayesh Bondu, Samyak Datta, Sara Chugh, Sara Hunt, Sargun Dhillon, Sasha Sidorov, Satadru Pan, Saurabh Mahajan, Saurabh Verma, Seiji Yamamoto, Sharadh Ramaswamy, Shaun Lindsay, Shaun Lindsay, Sheng Feng, Shenghao Lin, Shengxin Cindy Zha, Shishir Patil, Shiva Shankar, Shuqiang Zhang, Shuqiang Zhang, Sinong Wang, Sneha Agarwal, Soji Sajuyigbe, Soumith Chintala, Stephanie Max, Stephen Chen, Steve Kehoe, Steve Satterfield, Sudarshan Govindaprasad, Sumit Gupta, Summer Deng, Sungmin Cho, Sunny Virk, Suraj Subramanian, Sy Choudhury, Sydney Goldman, Tal Remez, Tamar Glaser, Tamara Best, Thilo Koehler, Thomas Robinson, Tianhe Li, Tianjun Zhang, Tim Matthews, Timothy Chou, Tzook Shaked, Varun Vontimitta, Victoria Ajayi, Victoria Montanez, Vijai Mohan, Vinay Satish Kumar, Vishal Mangla, Vlad Ionescu, Vlad Poenaru, Vlad Tiberiu Mihailescu, Vladimir Ivanov, Wei Li, Wenchen Wang, Wenwen Jiang, Wes Bouaziz, Will Constable, Xiaocheng Tang, Xiaojian Wu, Xiaolan Wang, Xilun Wu, Xinbo Gao, Yaniv Kleinman, Yanjun Chen, Ye Hu, Ye Jia, Ye Qi, Yenda Li, Yilin Zhang, Ying Zhang, Yossi Adi, Youngjin Nam, Yu, Wang, Yu Zhao, Yuchen Hao, Yundi Qian, Yunlu Li, Yuzi He, Zach Rait, Zachary DeVito, Zef Rosnbrick, Zhaoduo Wen, Zhenyu Yang, Zhiwei Zhao, and Zhiyu Ma. The llama 3 herd of models, 2024. URL `https://arxiv.org/abs/2407.21783`.

[7] Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. The dawn of natural language to sql: Are we fully ready? *Proceedings of the VLDB Endowment*, 17(11):3318–3331, July 2024. ISSN 2150-8097. doi: 10.14778/3681954.3682003. URL `http://dx.doi.org/10.14778/3681954.3682003`.

[8] Haoyang Li, Jing Zhang, Cuiping Li, and Hong Chen. Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql, 2023. URL `https://arxiv.org/abs/2302.05965`.

[9] OpenAI, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mohammad Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus,

Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Jan Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O'Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr H. Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine B. Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2024. URL `https://arxiv.org/abs/2303.08774`.

[10] Simone Papicchio, Simone Rossi, Luca Cagliero, and Paolo Papotti. Think2sql: Reinforce llm reasoning capabilities for text2sql, 2025. URL `https://arxiv.org/abs/2504.15077`.

[11] 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`.

[12] Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O. Arik. Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql, 2024. URL `https://arxiv.org/abs/2410.01943`.

[13] Qwen, :, An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, Huan Lin, Jian Yang, Jianhong Tu, Jianwei Zhang, Jianxin Yang, Jiaxi Yang, Jingren Zhou, Junyang Lin, Kai Dang, Keming Lu, Keqin Bao, Kexin Yang, Le Yu, Mei Li, Mingfeng Xue, Pei Zhang, Qin Zhu, Rui Men, Runji Lin, Tianhao Li, Tianyi Tang, Tingyu Xia, Xingzhang Ren, Xuancheng Ren, Yang Fan, Yang Su, Yichang Zhang, Yu Wan, Yuqiong Liu, Zeyu Cui, Zhenru Zhang, and Zihan Qiu. Qwen2.5 technical report, 2025. URL `https://arxiv.org/abs/2412.15115`.

[14] Zhihui Shao, Shubin Cai, Rongsheng Lin, and Zhong Ming. Enhancing text-to-SQL with question classification and multi-agent collaboration. In Luis Chiruzzo, Alan Ritter, and Lu Wang, editors, *Findings of the Association for Computational Linguistics: NAACL 2025*, pages 4340–4349, Albuquerque, New Mexico, April 2025. Association for Computational

Linguistics. ISBN 979-8-89176-195-7. doi: 10.18653/v1/2025.findings-naacl.245. URL https://aclanthology.org/2025.findings-naacl.245/.

[15] Jiawei Shen, Chengcheng Wan, Ruoyi Qiao, Jiazhen Zou, Hang Xu, Yuchen Shao, Yueling Zhang, Weikai Miao, and Geguang Pu. A study of in-context-learning-based text-to-sql errors, 2025. URL https://arxiv.org/abs/2501.09310.

[16] Liang Shi, Zhengju Tang, Nan Zhang, Xiaotong Zhang, and Zhi Yang. A survey on employing large language models for text-to-sql tasks. *ACM Comput. Surv.*, June 2025. ISSN 0360-0300. doi: 10.1145/3737873. URL https://doi.org/10.1145/3737873. Just Accepted.

[17] Noah Shinn, Federico Cassano, Edward Berman, Ashwin Gopinath, Karthik Narasimhan, and Shunyu Yao. Reflexion: Language agents with verbal reinforcement learning, 2023. URL https://arxiv.org/abs/2303.11366.

[18] SQLite. URL https://github.com/sqlite/sqlite.

[19] Chang-You Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. Exploring chain-of-thought style prompting for text-to-sql, 2023. URL https://arxiv.org/abs/2305.14215.

[20] Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, LinZheng Chai, Zhao Yan, Qian-Wen Zhang, Di Yin, Xing Sun, and Zhoujun Li. MAC-SQL: A multi-agent collaborative framework for text-to-SQL. In Owen Rambow, Leo Wanner, Marianna Apidianaki, Hend Al-Khalifa, Barbara Di Eugenio, and Steven Schockaert, editors, *Proceedings of the 31st International Conference on Computational Linguistics*, pages 540–557, Abu Dhabi, UAE, January 2025. Association for Computational Linguistics. URL https://aclanthology.org/2025.coling-main.36/.

[21] Zhongyuan Wang, Richong Zhang, Zhijie Nie, and Jaein Kim. Tool-assisted agent on sql inspection and refinement in real-world scenarios, 2024. URL https://arxiv.org/abs/2408.16991.

[22] 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, 2019. URL https://arxiv.org/abs/1809.08887.

[23] Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. Act-sql: In-context learning for text-to-sql with automatically-generated chain-of-thought, 2023. URL https://arxiv.org/abs/2310.17342.