AutoRuleSQL: Hybrid Text-to-SQL via Rule-Driven Fast Paths and LLM Bootstrapping

Han Xu
Univ. of Illinois at Urbana-Champaign
Urbana, IL, USA
hanxu8@illinois.edu

Yang Li
William & Mary
Williamsburg, VA, USA
yli102@wm.edu

Yanhai Xiong
William & Mary
Williamsburg, VA, USA
yxiong05@wm.edu

Robert Mintern
James City County Fire Department
Williamsburg, VA, USA
robert.mintern@jamescitycountyva.gov

Amir Louka
VCU Health
Williamsburg, VA, USA
amir.louka@vcuhealth.org

Haipeng Chen
William & Mary
Williamsburg, VA, USA
hchen23@wm.edu

Abstract

Natural Language to SQL (NL2SQL) enables natural language access to structured data, but LLM-based methods can be inefficient for real-time use and repetitive query patterns. We present AutoRuleSQL, a hybrid system that combines template-based fast paths with LLM fallback and offline bootstrapping. Empirical results show that it reduces latency by over 12.6% and improves execution accuracy by up to 4.0%, when combined with existing NL2SQL methods.

ACM Reference Format:
Han Xu, Yang Li, Yanhai Xiong, Robert Mintern, Amir Louka, and Haipeng Chen. 2025. AutoRuleSQL: Hybrid Text-to-SQL via Rule-Driven Fast Paths and LLM Bootstrapping. In *Proceedings of the 34th ACM International Conference on Information and Knowledge Management (CIKM '25), November 10–14, 2025, Seoul, Republic of Korea.* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/3746252.3761438

1 Introduction

Natural language to SQL (NL2SQL) allows non-technical users to query structured data using plain language [2]. These systems translate natural language questions into executable SQL queries, lowering barriers to data access for domains such as healthcare, public services, and business analytics. Recent advancements in large language models (LLMs) have significantly improved NL2SQL performance. Benchmarks such as Spider [8] and BIRD [4] show that these models can handle complex schemas and generate accurate SQL queries.

However, LLM-based solutions are often slow and costly for real-time deployment, particularly in high-frequency query environments like clinical dashboards or municipal portals. These systems often rely on multiple LLM calls for schema linking, candidate generation, and reranking, leading to increased latency and computational cost [6]. Moreover, many such queries follow recurring patterns, making full LLM inference unnecessary.

CIKM '25, Seoul, Republic of Korea.

ACM ISBN 979-8-4007-2040-6/2025/11
https://doi.org/10.1145/3746252.3761438

We propose AutoRuleSQL, a hybrid framework that combines fast rule-based query resolution with asynchronous LLM-based template bootstrapping. Incoming queries are matched against a template library; if matched, SQL is generated instantly. Otherwise, the system falls back to an LLM specialized in NL2SQL. Validated outputs from the LLM are later used to create new templates offline.

This design enables low-latency query resolution while continuously expanding coverage. On the Spider test set, it reduces query latency by over 12.6% and improves execution accuracy (EA) by up to 4.0%, demonstrating its effectiveness for real-time deployment.

2 Methodology

We propose an adaptive hybrid framework that consists of two primary components: a real-time query processing engine and an offline template generation module. The overall architecture is illustrated in Figure 1.

2.1 Real-time Query Processing

At runtime, each incoming natural language query (NLQ) is matched against a set of predefined templates. Each template contains:

(1) A pattern with named capture groups (e.g., datetime, topk).
(2) A corresponding SQL template with placeholders to be filled using the capture values.

When a query matches a template, the captured slot values are inserted into the SQL template to generate the final SQL query. Templates are designed to cover commonly observed query categories such as date filters, top-k, aggregation, and comparison.

If no template matches the incoming NLQ, the query is forwarded to a fallback NL2SQL component—typically a specialized LLM (e.g., OmniSQL [3]) or an NL2SQL framework (e.g., DIN-SQL [5] and CHESS [7])—to generate the corresponding SQL query.

2.2 Offline Template Generation

As a continuation of the fallback path described in Section 2.1, this module enables the system to learn from validated LLM outputs and expand its template library. When a new (NLQ, SQL) pair is successfully validated—through correct execution and optional user approval—it is added to a queue for offline template induction.

During system idle periods, a background worker first identifies the query category associated with the (NLQ, SQL) pair. It then prompts a template LLM—either a proprietary model or an

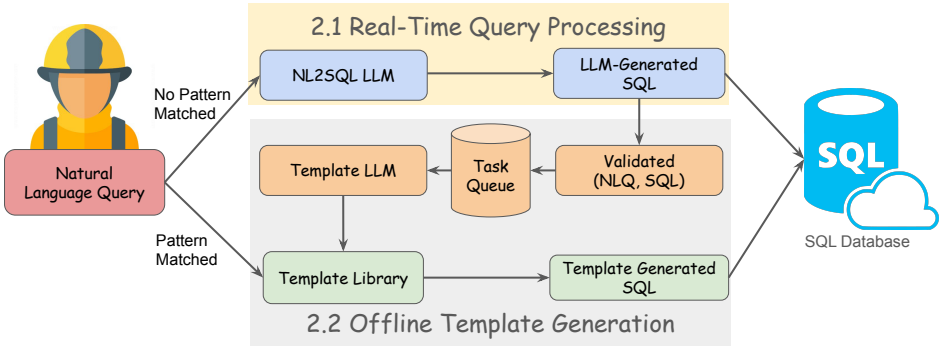

Figure 1: An illustration of the AutoRuleSQL hybrid workflow.

open-source alternative—with category-specific instructions to generate a reusable template, consisting of a matching pattern and a parameterized SQL template (see Section 2.1).

To ensure correctness and consistency, a newly generated template is accepted into the template library only if its output SQL is validated—specifically, it must be structurally equivalent to the original SQL via normalization, and the execution results must be identical. Once validated, the template is added to the library, enabling the system to handle similar future queries instantly without invoking the LLM.

The background worker runs at a lower priority than the real-time query engine and is designed to pause whenever the system is actively processing user queries. To manage memory and lookup efficiency, the template library maintains a fixed-size cache using a least frequently used (LFU) replacement policy, retaining only the top-k most accessed templates over time.

3 Experiments

Using GPT-4.1 on the Spider test set, AutoRuleSQL converted 25.5% of queries into reusable templates with an LFU cache of size k = 1000.

To evaluate generalization, we modified the Spider test set by replacing slot values in template-matched queries with unseen ones. We compared AutoRuleSQL against two baselines: OmniSQL-7B (bfloat 16, using greedy search), a specialized NL2SQL language model, and DIN-SQL, an NL2SQL framework paired with GPT-4.1. All methods were evaluated on SQL generation time and EA, using an AMD EPYC 7532 CPU and an Nvidia A100 40G GPU.

Table 1: Comparison of SQL generation speed and EA.

Method	Avg Time (s)	EA (%)
OmniSQL-7B	0.20	87.3
OmniSQL-7B + AutoRuleSQL	**0.19**	**88.6**
DIN-SQL (GPT-4.1)	5.54	79.5
DIN-SQL (GPT-4.1) + AutoRuleSQL	4.84	82.7

Table 1 shows that the AutoRuleSQL integration improves both latency and EA. With OmniSQL-7B, it reduces average generation time from 0.20s to 0.19s and increases EA from 87.3% to 88.6%. With DIN-SQL, it reduces time from 5.54s to 4.84s and raises EA from 79.5% to 82.7%.

In addition to these improvements, AutoRuleSQL handles frequent and previously seen queries through lightweight templates, which execute on CPU without invoking LLM inference. This enables more efficient CPU–GPU resource allocation.

4 Conclusion

AutoRuleSQL combines rule-based matching with LLM fallback and offline template bootstrapping to reduce latency and improve efficiency in NL2SQL tasks. It adapts over time by learning new templates from validated queries, achieving higher accuracy with lower response times. Future research can further adapt this approach to domains like log analysis and system monitoring [1].

5 Speaker Bio

Han Xu is a graduate student from the University of Illinois Urbana-Champaign. His expertise spans AI and ML.

Acknowledgments

This work was supported in part by the National Science Foundation (NSF) through the NAIRR Pilot under request number NAIRR250123. Generative AI (GenAI) was used to assist with code development; all code was reviewed by the authors.

References

[1] Cheng Ji et al. 2025. Leveraging Large Language Model for Intelligent Log Processing and Autonomous Debugging in Cloud AI Platforms. *arXiv preprint arXiv:2506.17900* (2025).
[2] Hyeonji Kim et al. 2020. Natural language to SQL: Where are we today? *Proceedings of the VLDB Endowment* 13, 10 (2020), 1737–1750.
[3] Haoyang Li et al. 2025. Omnisql: Synthesizing high-quality text-to-sql data at scale. *arXiv preprint arXiv:2503.02240* (2025).
[4] Jinyang Li et al. 2023. Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls. *Advances in Neural Information Processing Systems* 36 (2023), 42330–42357.
[5] Mohammadreza Pourreza et al. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. *Advances in Neural Information Processing Systems* 36 (2023), 36339–36348.
[6] Liang Shi et al. 2024. A survey on employing large language models for text-to-sql tasks. (2024).
[7] Shayan Talaei et al. 2024. Chess: Contextual harnessing for efficient sql synthesis. *arXiv preprint arXiv:2405.16755* (2024).
[8] Tao Yu et al. 2018. Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *arXiv preprint arXiv:1809.08887* (2018).