Converting Natural Language to Query Languages Using Large Language Models

A Systematic Literature Review

Rayfran Rocha Lima
rayfran.lima@sidia.com
Sidia Institute of Technology
Manaus, AM, Brazil

Kamila Cardoso Vasconcelos
kamila.vasconcelos@sidia.com
Sidia Institute of Technology
Manaus, AM, Brazil

Eloisa Mendonça Gadelha
eloisa.gadelha@sidia.com
Sidia Institute of Technology
Manaus, Brazil

ABSTRACT

The task of converting natural language commands into query languages (NL-to-QL) has gained attention due to its potential to improve data accessibility for non-technical users. Although several studies have explored rule-based and sequence-to-sequence models, there is still a lack of a literature review that presents the impact of using large language models (LLMs) on this task. As an output of a systematic literature review, this paper examines how recent studies have utilized LLMs by applying fine-tuning or prompt engineering techniques to address this task. Presenting a compilation of methods, architectures, and techniques, as well as evaluation metrics, datasets, and benchmarks applied, including the available competitions and educational platforms, it provides a comprehensive overview of NL-to-QL conversion, mapping current advancements, future research directions, and remaining challenges, including issues with schema generalization, query interpretability, and hallucination mitigation.

KEYWORDS

Txt2SQL, NL-to-QL, NL Query to Query Language

1 INTRODUCTION

Although many companies offer Business Intelligence (BI) platforms provide user-friendly dashboards and reports [82], they often lack the flexibility to meet users' specific needs [45]. To access detailed data or data that goes beyond the scope of BI systems, users must learn and manually construct query languages, such as structured query language (SQL) for relational databases or proprietary query languages (PQL) for non-relational databases, which can be daunting for those without technical expertise. The emergence of diverse database systems further complicates this issue, as each system may require different query languages or API-based libraries.

The task of converting natural language into query languages (NL-to-QL) has been explored as a means to bridge the gap between users and data [79]. Over the decades, approaches have evolved from rule-based systems [14, 60] and syntactic parsing methods to deep learning [73, 90, 95] and transformer-based models [52, 56, 61, 75]. Recent advances in the use of pre-trained transformer models (e.g., ChatGPT, LLAMA, DeepSeek) combined with in-context learning techniques [59, 62, 69] have significantly improved accuracy and generalization, enabling applications in a wide range of domains [35]. Nevertheless, challenges remain, including the interpretation of ambiguous queries [28, 56], adaptation to heterogeneous database schemas [49, 64], cross-domain generalization [3, 11, 36], and the generation of efficient queries for complex information needs [61, 74].

Given the continuous evolution and diversity of NL-to-QL conversion approaches, recent studies have surveyed aspects such as encoding and decoding techniques [6, 16], the evolution of applied methods [17, 63], evaluation metrics, dataset benchmarks, and system architectures [37, 94]. However, no single study has consolidated the available approaches, technologies, or techniques across the entire NL-to-QL conversion pipeline—from input methods and human-computer interaction interfaces to natural language understanding, metadata mapping, query generation, result analysis, and model robustness across domains.

To address this gap, this paper presents the results of a systematic literature review aimed at mapping the current SOTA in NL-to-QL conversion. The review identifies key developments, ongoing challenges, emerging research trends, and offers strategic recommendations for future research directions in this domain.

2 BACKGROUND

Query languages are specialized languages designed to access and manipulate data in databases [77]. These databases can be relational, non-relational, or distributed, and their corresponding query languages follow specific syntax and structural rules [41]. Although SQL is the most widely adopted language for general database queries, enterprises also employ domain-specific languages in their data ecosystem such as Jira Query Language (JQL) for Jira, Cassandra Query Language (CQL) for Apache Cassandra, and MongoDB Query Language (MQL) for MongoDB [46, 80, 83].

Despite their effectiveness, these languages present challenges for users lacking technical expertise. Query formulation requires an understanding of database schemas, relationships, and syntax, which can be complex and unintuitive [1]. To bridge this gap, researchers have developed NL interfaces that translate user queries into structured QL, enhancing accessibility [21, 79, 88].

The evolution of NL-to-QL conversion has been driven by advancements in natural language processing and machine learning. Early rule-based approaches relied on handcrafted templates and syntactic rules, but they lacked scalability and generalization [14]. Neural network-based models later improved query generation through sequence-to-sequence (Seq2Seq) architectures [53, 95].

In: Proceedings of the Brazilian Symposium on Multimedia and the Web (WebMedia'2025). Rio de Janeiro, Brazil. Porto Alegre: Brazilian Computer Society, 2025.
© 2025 SBC – Brazilian Computing Society.
ISSN 2966-2753

Further refinements introduced encoders based on sequence modeling [88, 89], graph-based representations [9, 85], and decoders employing structure-based [13, 50], grammar-based [22, 27], and execution-based [33, 73] SQL generation. Encoder-decoder architectures have significantly enhanced SQL generation accuracy and adaptability [25].

More recently, foundation models (FM-based models), such as Generative Pre-trained Transformer (GPT), Bidirectional Encoder Representations from Transformers (BERT), and Text-To-Text Transfer Transformer (T5), have demonstrated exceptional capabilities in performing complex tasks with minimal retraining [74]. While BERT and T5 [52] focus primarily on understanding and transforming text through bidirectional and unified text-to-text approaches, respectively, GPT models like ChatGPT, Llama, and DeepSeek, support diverse applications, including text generation, summarization, classification, translation, and NL-to-QL conversion [47, 84].

Datasets. Datasets are fundamental in training and evaluating NL-to-QL models. These datasets, which can be public or proprietary, are built from extensive text collections extracted from web pages, private systems, or structured sources and are commonly stored in XML format [24]. Text-based datasets are particularly crucial for this task, as they provide linguistic diversity for model training.

Some datasets are general-purpose, such as SQLWiki [33], which contains SQL queries from Wikipedia. Others are domain-specific, such as ATIS [29] and FIBEN [62], which contain specialized queries from aviation and financial domains, respectively. Recently, cross-domain datasets have gained prominence for evaluating model generalization. Benchmarks like WikiSQL and Spider [43, 90] are widely used to assess performance across multiple database schemas. These resources are key for testing model robustness against schema variability and query complexity.

Related Works. The conversion of NL commands into SQL has been extensively studied, particularly in the context of deep learning and natural language interfaces for databases. Several systematic reviews have mapped advancements in the field, identifying challenges and emerging trends [6, 16, 37, 38, 63]. However, gaps remain that must be addressed to consolidate this technology further. These studies have examined different approaches for NL-to-QL conversion, focusing on deep learning, model generalization, and analysis techniques, but none has evaluated the impact of using LLM in performing this task. Deng et al. [16] reviewed advances in semantic encoding and decoding but did not explore the role of Large Language Models (LLMs). Qin et al. [63] classified approaches based on seq2seq and transformers but did not investigate modern prompt engineering techniques. Kanburoğlu and Tek [37] highlighted challenges in adapting to different database schemas, while Baig et al. [6] emphasized the need for improved model interpretability. Katsogiannis-Meimarakis and Koutrika [38] analyzed evaluation challenges but did not address LLM-specific metrics.

While the existing reviews provide valuable insights and data for addressing the task of converting natural language (NL) to query languages (QL), they lack a systematic investigation of the role of Large Language Models (LLMs) in this domain. Specifically, these works do not explore how prompting engineering or fine-tuning techniques can enhance NL-to-QL conversion, nor do they examine the diversity of query languages, the variety of conversion methods, or the key challenges and evaluation metrics employed in this field. This gap highlights the necessity of a comprehensive review that focuses on LLMs and their applications in NL-to-QL conversion, as well as the identification of remaining research gaps. This study aims to fill this void by offering a structured analysis of the current state of the art, providing a foundation for future advancements in this rapidly evolving area.

3 METHOD

To consolidate the state-of-the-art from existing empirical studies, a systematic literature review [40] was conducted to answer three research questions:

RQ1. What methods and approaches are applied?
RQ2. What are the main limitations and challenges?
RQ3. How are the results evaluated?

While RQ1 addresses the main techniques employed to resolve this task, RQ2 explores the limitations and challenges faced by researchers, and RQ3 reveals the patterns of metrics and benchmarks used to evaluate the precision and efficiency of converting NL commands to query language commands.

After several runs to reduce the noise in the results while ensuring that the reference set, composed of [56, 61, 75, 91], was retrieved, we set the final Search String as *(LLM OR "Large Language Model" OR GPT OR "Generative Pretrained Transformer" OR "Generative AI" OR "GenAI" OR ChatGPT OR "Artificial Inteligence" OR AI OR "AI-Assisted" OR "AI-Based") AND ("convert" OR "conversion" OR "txt-to" OR "Text-to" OR "NL-to") AND ("Query Language" OR SQL OR JQL OR CQL OR MQL OR GraphQL OR QL)*. Applied in the Scopus [1], WebOfScience [2], IEEE [3], ACM [4] in October 2024, after retrieving the entire result with an additional publication year filter from 2020 to 2024, we got 379 candidate papers.

The study selection process involved two filtering steps, each applying the inclusion and exclusion criteria. To be included in the selected list, the candidate study must: (i) address the use of LLM to convert Natural Language commands to executable QL, (ii) be a primary study [20], (iii) be published between 2020 and 2024, (iv) be peer-reviewed and published in a journal or conference, and (v) be written in English. In contrast, candidate study must be excluded if it: (i) does not provide any answer to the research questions of this study or (ii) lack a way to get access to the paper.

In Filter 1, the result of the analysis of the title and abstract of the retrieved papers reduced the set of candidate studies to 46 (12% of the total). To ensure consistency in paper selection, we applied statistical tests to evaluate inter-rater agreement where two independent reviewers screened articles based on predefined criteria, and any discrepancies were resolved through a reconciliation meeting with an additional researcher serving as a mediator. We calculated Cohen's Kappa coefficient to quantify agreement, with a threshold of 0.83 indicating strong agreement between raters [42].

In Filter 2, we did a full read of each cadidate studies to extract data to support answering the research questions and reapply the inclusion and exclusion criteria which supported to identified 13

[1]http://www.digitallibrary.edu.pk/scopus.html
[2]https://www.webofscience.com/wos/
[3]https://ieeexplore.ieee.org/
[4]https://dl.acm.org/

publications that were removed from the selected list resulting in the final selection of 33 papers divided by applied Approach, as shown in Table 1.

Table 1: List of selected papers

Approach	Selected papers	Total
Fine-tuning	[2, 23, 36, 52, 91]	5 (15%)
Prompt engineering	[3, 5, 10, 11, 25, 26, 28, 44, 49, 55–57, 59, 61, 62, 64, 67–70, 74–76, 78, 79, 81, 93, 95]	28 (85%)

Afterward, we implemented data analysis, involving content analysis [30], synthesis using thematic analysis [15], and critical evaluation of selected papers to derive meaningful insights and results. The thematic synthesis approach involved extracting key data from each secondary study (refer Table 2) and grouping similar elements into preliminary categories. These categories were iteratively refined by two researchers through discussions and validation against the study objectives. A third researcher reviewed the final categories to ensure consistency and minimize bias. This process ensured that the synthesized factors were representative and aligned with the research questions.

Table 2: Questions to data extraction

Questions
1. What is the motivation and context?
2. What are the limitations of related works?
3. What is the objective of the paper?
4. What types of text inputs were used as sources for QL command conversion?
5. What interface was used for input?
6. What LLMs, encoders, and decoders were employed?
7. What datasets or benchmarks were used?
8. Was any competition platform mentioned?
9. How were the results measured?
10. What are the common challenges, pitfalls, or limitations?
11. What domains were applied?
12. How issues like ambiguity, uncertainty, and noise were handled?
13. What is the complexity of QL generated?
14. How entity recognition (e.g. tables, values) were applied?
15. What are advantages and disadvantages reported on using LLMs?
16. Did the study attempt to leverage knowledge from one domain to another?
17. If engineering of prompts was used, what type of technique was employed?
18. If the paper mentions model training (fine-tuning), how was it implemented?
19. How was limited/missing/unavailable input data addressed?
20. What are the main findings and takeaways?
21. What does the paper propose for future work?

4 RESULTS

This section presents the results and findings addressing the three research questions. Section 4.1 examines methods used for converting natural language into executable query languages (RQ1). Section 4.2 outlines the main challenges faced in this process (RQ2). Finally, Section 4.3 reviews evaluation metrics and benchmarks used to assess the performance of these conversion models (RQ3).

4.1 RQ1. What methods and approaches are applied?

RQ1 investigates the methods and approaches applied in converting natural language to query language commands, focusing on the motivations and objectives of the studies, the types of standardized text inputs and input interfaces used, as well as the LLMs and datasets employed. It also explores the domains where these conversions are commonly applied, the reuse of learning across domains, the prompt engineering techniques utilized or LLM training approaches applied.

Focus and Deliverables. The analyzed studies were categorized into four groups based on their objectives: (i) *Performance Enhancement* includes works that focus on improving the efficiency and accuracy of NL-to-QL conversion through prompt engineering techniques [49, 56, 62, 68, 70, 75, 76, 78, 93] and model training [2, 23, 41, 52, 61]; (ii) *Proposals of New Methods or Models* features contributions developing novel frameworks and algorithms to address field-specific challenges [2, 49, 59, 64]; (iii) *Creation of Metrics and Evaluation* includes studies which proposed new metrics to assess model effectiveness [28, 61, 71]; (iv) *Miscellaneous* category encompasses diverse studies focusing on query completeness verification [52], others comparing zero-shot, few-shot [3, 55, 62, 75, 76, 92], and supervised fine-tuning approaches [2, 23, 69].

Inputs and interfaces. In experiments aimed at addressing the task of converting natural language commands into QL commands researchers have employed a variety of inputs and interfaces to optimize performance and accuracy. Inputs predominantly consist of natural language queries [71, 78, 91] in free-form or structed text that serves as the primary source for QL conversion. Specific domain-driven inputs are utilized, where complex queries are derived from specialized datasets, such as the Presidential Database [61]. There were strategies that used natural language questions, database schemas, and data examples as inputs [57]. This approach allows researchers to assess models' capabilities in handling nuanced, context-specific queries.

Regarding interfaces, prompt-based mechanisms are the most prevalent [10, 71, 91]. Some studies used programming libraries, such as LangChain framework[5], to input prompt instructions directly to LLMs [56]. Some studies built proprietary systems (eg. Querylizer) that served as an interface between users and the conversion system supported by LLMs [18]. These approaches collectively contribute to enhance the way users interact with systems that are proposed to support converting NL commands into QL commands.

Query Languages. Although terms such as JQL, CQL, MQL, and GraphQL were included in the Search String, we observed a significant predominance of efforts directed at solving the NL-to-SQL conversion task, since 93% of the analyzed studies used SQL as the main language. Only Ni et al. [58] worked in task that converts the user's questions into GraphQL that is a query language and API runtime that is revolutionizing the way developers build and interact with application programming interfaces.

Bathla et al. [7] explored an AI-driven approach to transform SQL commands into programmable code compatible with NoSQL databases. Their focus included languages such as: (i) Cassandra Query Language (CQL), designed for column-oriented data storage and used in Cassandra applications to manage both structured and semi-structured data; (ii) Unstructured Query Language (UnQL), specifically tailored for Couchbase databases; (iii) Cypher, a graph-based language initially developed for Neo4j, optimized for processing graph-based databases; (iv) MongoDB Query Language, which

[5]https://www.langchain.com/

differs from SQL queries by utilizing collection-based queries to interact with MongoDB.

Pipeline. In the task of converting NL commands to QL queries using LLMs, regardless of the pipeline of adopted actions, common steps used in selected studies include:

Step 1: Preprocessing the input, including normalization and cleaning of the text (stopword removal, stemming, tokenization, etc.), identification of relevant keywords (table names, columns, operators), and parsing the grammatical structure of the sentence to extract entities and relations.

Step 2: Representing the input with natural language transformation in a format suitable for the model (tokens, embeddings, vector representations). If applicable, incorporation of information from the database schema (e.g., existing tables and columns).

Step 3. NL-to-QL mapping. If using a seq2seq or Transformer model, the model generates the corresponding QL sequence. If using a pipeline based on rules, heuristics or template searches, the elements of the NL are matched to the correct QL syntax. In the case of a hybrid approach, a first model can structure the query and a second refine it.

Step 4. Validation and post-processing, which may include syntactic and semantic verification of the generated QL query, automatic error correction (such as adding necessary JOINs, filling in missing WHEREs), and refinement based on specific rules of the database schema.

Step 5. Execution and interactive refinement, where tests are run on the generated QL query on a fictitious or real database to ensure that it is valid. If applicable, iterative refinement with user feedback or a rewrite model can be added.

Step 6. Continuous learning and improvement. In this step, the model can be adjusted with new examples to improve accuracy or incorporate feedback from malformed or inadequate queries to refine the approach.

Each architecture may handle these steps differently, but all should consider this general flow to ensure accurate and efficient NL-to-QL conversion.

Architecture. Using large-scale language models, such as ChatGPT [28, 41, 76] or LLama [71, 75], which generate QL from natural language based on massive pre-training data, these models can be used with prompt engineering or fine-tuning to adapt them to specific tasks [2, 91]. The benefits of this approach include the ability to integrate easily into interactive systems without requiring domain-specific training, as well as the potential for high-precision results. However, it is not without its challenges: these models can generate incorrect QL queries if not properly validated, and they often require explicit contextual information to achieve optimal precision.

The Table 3 demonstrate the popularity of the models used in task of converting NL-to-QL. While some studies rely solely on LLMs like ChatGPT [1, 36, 49, 59, 71, 95] or the Llama [23, 59, 71] family for this task, others employ ensembling strategies where LLMs are used as both encoders and decoders to enhances input processing and ensures accurate query generation [11, 55, 69, 96]. Most of the prompt engineering-based studies end up using more than one LLMs to compare their capabilities and limitations [49, 67, 75, 78].

Datasets. The analyzed NL-to-QL experiments utilized a wide variety of datasets, reflecting the diversity and complexity of the task. Table 4 shows that the Spider [90] is the most frequent dataset used in the experiments, mentioned in 70% of the selected studies [2, 3, 11, 23, 26, 52, 56, 57, 61, 62, 69, 74–76, 78, 79, 81, 91].

Table 3: Popularity of LLMs applied in selected studies

Models applied in the studies	% Mentioned
GPT-3.5-Turbo	62.0%
GPT-4	48.0%
LLaMa-3	17.0%
Codex	14.0%
ALPACA, BART, RASAT, MPT, Bard, Vicuna, CODE-DAVINCI-002/003	7.0%
COPILOT, T5, MT5, Code Llama, DIN-PRO, C3, CodeS-15B, CodeS-7B, Dolly, Guanaco	3.0%

Other widely used datasets include WikiSQL (10%) used in [10, 36, 52, 70], KaggleDBQA (13%) [3, 11, 69], followed by BIRD (10%) [44, 56, 57, 62], ATIS (7%) [58, 70] and Geo-Query (7%) [58, 78, 91]. Claiming that the QL generation task on WikiSQL is basically solved due to simplicity of its data, Zhang et al. [95] used a more complicated dataset called TableQA composed of 45,918 query-sql pairs.

Table 4: Frequency of Dataset Usage in NL-to-QL Conversion Experiments

Dataset	#Databases	#Tables per Database	#NL-QL pairs	QL query level	% Mention
Spider	200	On average 5	10,181	Complex	70.0%
KaggleDBQA	444K	-	-	-	13.0%
WikiSQL	24241	1	80,654	Simple	10.0%
BIRD	95	-	12,751	-	10.0%
ATIS	1	32	5,280	Complex (no HAVING and ORDER BY)	7.0%
GeoQuery	1	6	877	Complex	7.0%

We identified other datasets, but these did not recur, such as Hospital patient database[52], Finance [52, 70, 91], BULL[91], CoSQL [36], DBs related to drugs, sport center, COVID19 and HybridQA [49, 74], GitHub repository [81], MTOP and MCWQ-MCD3 [96], OncoMX [93], Presidential Database [44], RAT-SQL and RESDSQL [92], Scholar and Restaurants [78], TUR2SQL[36]. This suggests that despite the variety of datasets, many solutions remain limited to specific contexts, restricting the generalizability of the results.

Domains. The application of NL-to-QL models has been explored across various domain-specific contexts, demonstrating both their potential and limitations. In business intelligence, these models facilitate integration with visualization tools [18], while in customer support, they manage ambiguous, multi-domain queries [28]. E-commerce platforms leverage these models for efficient post-training query generation [3], and the healthcare sector uses them for structured patient data management [52]. In finance, models assist with fraud detection and investment analysis using machine learning techniques [52]. Furthermore, their cross-lingual and cross-database capabilities have been demonstrated, broadening their applicability [69]. However, despite their versatility, many models remain confined to specific domains, limiting their generalizability across diverse datasets and applications.

Attempts to reuse models across domains face significant challenges due to the dynamic and unpredictable nature of database schemas. While models effectively apply query syntax and generate structurally correct QL, predicting table names, fields, and values remains problematic, as these elements are tied to specific domain knowledge. This issue is exacerbated by domain shift, where models trained in areas like nuclear data or crime statistics perform poorly in unrelated contexts [66]. Techniques such as domain adaptation, prompt engineering, and fine-tuning offer partial solutions [51], but models still lack the robustness to handle schema variability. Without explicit information on database structure, models often produce inaccurate or incomplete queries, highlighting the need for robust frameworks and external knowledge integration to enhance cross-domain adaptability.

QL commands complexity. The findings over this subject suggests that the complexity of QL queries generated or tested in experiments varies significantly across studies. While some studies used WikiSQL dataset that contains relatively simple queries, typically involving single tables and basic commands [58], others dealt with multiple tables and relationships of a proprietary Presidential database [61] having to deal with multiple joins, subqueries, and aggregations. Although there is variety among studies, the type of NL queries and their QL command (see Table 5) adapted from BIRD dataset [44] showcases the range of options and complexity involved in addressing the convertion of NL to QL task.

Data recognition and extraction from NL commands. One of the most fundamental problems in natural language processing, seeks to identify the boundaries and types of entities with specific meanings in natural language text [48]. Recent studies have explored various approaches for recognizing entities, values, and data types[32, 39]. While large language models (LLMs), such as ChatGPT, have demonstrated impressive capabilities in extracting data from natural language commands without the need for fine-tuning [78], these models still exhibit limitations when it comes to precisely identifying tables, columns, and constraints compared to models specifically fine-tuned for this task [52]. Additionally, research has investigated the impact of prompt engineering techniques, where composed prompts consisting of database schema, real data examples, and additional orientation have significantly improved the recognition of entities, values, and data types [49].

LLM Training. Only 15% of selected studies adopt approaches to model fine-tuning. While some studies adopt regular fine-tuning [36, 52, 91], other applyed supervised fine-tuning (STF) [2, 23] as another option for training LLM for better NL-to-QL generational task.

For example, to do a regular fine-tune, Zhang et al. [91] used a database, called BULL, which contains data on the intelligent investment assistant product. The data were adjusted through the application of hybrid data augmentation, synonymous question augmentation, and rule-based augmentation. To overcome challenges such as high computational cost, high storage cost, and lack of cross-database generalization ability, they proposed a multi-task parameter-efficient finetuning method based on Low-Rank Adaptation (LoRA) [31] using open-source LLM such as LLaMa.

Contrasting, to do a SFT, Agrahari et al. [2] used Spider dataset [90]. As LLM they used Open Llama-V2 due to its several architectural advantages including pre-normalization, SwiGLU activation, and Rotary embeddings. Although they do not offer details of the fine-tuning process, the results pushes state of art accuracy on spider test suite to 89.6% on dev set which represent first instance of surpassing the earlier best-in-class with 5.5% higher score and 86.8% of exact match accuracy on dev set.

Table 5: Type of NL x QL commands

Category	NL command / SQL command example
Match-based	How many gas stations in CZE has Premium gas? SELECT COUNT(GasStationID) FROM gasstations WHERE Country = 'CZE' AND Segment = 'Premium'
Ranking	What are the titles of the top 5 posts with the highest popularity? SELECT Title FROM posts ORDER BY ViewCount DESC LIMIT 5
Comparison	How many color cards with no borders have been ranked higher than 12000 on EDHRec? SELECT COUNT(id) FROM cards WHERE edhrecRank > 12000 AND borderColor = 'borderless'
Counting	How many of the members' hometowns are from Maryland state? SELECT COUNT(T2.member_id) FROM zip_code AS T1 INNER JOIN member AS T2 ON T1.zip_code = T2.zip WHERE T1.state = 'Maryland'
Aggregation	What is the average height of the superheroes from Marvel Comics? SELECT AVG(T1.height_cm) FROM superhero AS T1 INNER JOIN publisher AS T2 ON T1.publisher_id = T2.id WHERE T2.publisher_name = 'Marvel Comics'
Domain Knowledge	Name all patient with hematoclit level exceeded the normal range. SELECT T1.name FROM Patient AS T1 INNER JOIN Laboratory AS T2 ON T1.ID = T2.ID WHERE T1.ID IN (SELECT ID FROM Laboratory WHERE HCT > 52)
Numeric Computation	Among the posts with a score of over 20, what is the percentage of them being owned by an elder user? SELECT CAST(SUM(IIF(T2.Age > 65, 1, 0)) AS REAL) * 100 / count(T1.Id) FROM posts AS T1 INNER JOIN users AS T2 ON T1.OwnerUserId = T2.Id WHERE T1.Score > 20
Synonym	How many clients opened their accounts in Jesenik branch were women? (female) SELECT COUNT(T1.client_id) FROM client AS T1 INNER JOIN district AS T2 ON T1.district_id = T2.district_id WHERE T1.gender = 'F' AND T2.A2 = 'Jesenik'
Value Illustration or Inference	Among the weekly issuance accounts, how many have a loan of under 200000? SELECT COUNT(T1.account_id) FROM loan AS T1 INNER JOIN account AS T2 ON T1.account_id = T2.account_id WHERE T2.frequency = 'POPLATEK TYDNE' AND T1.amount < 200000

Kanburoflu and Tek [36] created a Turkish dataset with data automatically generated with pairs of NL command and their respective QL which served of input to fine-tuning process. The about 11.000 records where divided in dev (10%), testing (20%) and training (70%) sub-datasets. They fine-tunned SQLNet model [88]. Its capacity to generate QL in this context was compared with ChatGPT models. The results shows that although the fine-tuning effort, ChatGPT achieved superior performance converting NL commands in SQL commands.

Prompt Engineering Techniques. 85% of the selected studies adopt prompt engineering to convert NL commands in QL commands. Using an alternative from fine-tuning, some studies employed various prompt engineering techniques such as zero-shot prompting [71], few-shot prompting strategy with the Structured and Unstructured Query Language (SUQL) method, incorporating specific examples to guide the model and improve accuracy in handling more complex queries [71], and chain of thoughts (CoT) prompting leveraging GPT-3.5 and GPT-4 to improve reasoning ability by producing intermediate steps before predicting a final answer [76].

Self-Consistency [93] mitigates the phenomenon of randomness in the output of LLMs by voting on the diversity of results and selecting the best one. Self-Debug [12] employing CoT prompting to obtain the question explanation and generates the initial QL, then instruct LLMs to debug the QL. Using a combination of prompting techniques, Zhao et al. (2024) [78] proposed three specific approaches for the NL-to-QL task: Schema Linking-prompt (SL), Clause by Clause-prompt (CC), and SL+CC prompt, each targeting distinct aspects of the conversion process, such as logical structuring and contextualization of commands.

While significant progress has been made in the development of NL-to-QL models, challenges related to generalizability, domain-specific limitations, and methodological inconsistencies persist. The diversity of datasets, approaches, and evaluation metrics demonstrates the field's richness but also highlights the need for greater standardization and robustness. Future research should focus on creating adaptable frameworks capable of addressing cross-domain variability and improving model performance in real-world applications.

4.2 RQ2. What are the main limitations and challenges?

RQ2 examines the primary limitations and challenges faced by the researchers, analyzing the constraints reported in related works, common pitfalls in using LLMs for NL-to-QL conversion, strategies for handling ambiguity, uncertainty, and noise in input data, and the comparative advantages and disadvantages of LLMs over traditional methods.

Limitations and Knowledge Gaps. Research on converting NL commands into SQL commands has made significant progress but still faces several limitations and knowledge gaps. Despite advancements, there is no practical benchmark for NL-to-QL tasks focused on financial analysis, and current methods fail to consider specific characteristics of these databases, such as wide tables [91]. While Tan et al. [78] points out that while LLMs have demonstrated impressive code generation capabilities without fine-tuning, their results remain inferior compared to models specifically fine-tuned for this task, contrasting this viewpoint, Kanburoflu and Tek [36] presents a fine-tunned model that generated inferior performance compared to ChatGPT.

Furthermore, model generalization continues to be a challenge, particularly in complex and heterogeneous domains. The lack of standardization in datasets and query evaluation complicates direct comparisons between different approaches, limiting progress toward scalable and reusable solutions [11]. Despite these challenges, there have been improvements in model efficiency, especially through the use of prompt engineering techniques and the adaptation of pre-trained models for specific tasks, paving the way for future research to address these limitations and broaden the applicability of proposed solutions.

Recurring Challenges and Pitfalls. Research exploring NL-to-QL conversion with LLMs identifies several recurring challenges, pitfalls, and limitations. Tan et al. [78] highlights the complexity of the schema linking task, which involves correctly identifying relevant column names in a query, such as associating "player_name" with the corresponding column in a table. This process is particularly challenging due to linguistic ambiguity and variations in database structures [91]. The work on DIN-SQL [62] proposed decomposing the NL-to-QL task into subtasks, including schema linking, classification, SQL generation, and self-correction, using techniques such as Chain of Thought prompt engineering technique. Although this approach has shown promising results, the authors acknowledge that solutions for these subtasks remain sub-optimal, particularly in the deep analysis of schema relationships and the accuracy of the generated queries. Furthermore, the reliance on specific training data limits the models' ability to generalize to distinct domains, and fine-tuning LLMs still faces challenges related to computational costs and the scarcity of high-quality annotated data.

Strategies for Addressing Ambiguity, Uncertainty, and Noise in Input Data. Studies on NL-to-QL conversion using LLMs face significant challenges related to ambiguity, uncertainty, and noise in input data. Zhang et al. [91] notes that LLMs exhibit uncertainty and instability due to the inherent randomness of these models, which can lead to different outputs for the same prompt. This behavior directly affects the consistency of the generated SQL queries, which may alternate between correct and incorrect results in subsequent executions. To mitigate this issue, there is a need to develop techniques that enhance the consistency of model outputs. Tan et al. [79] suggests that identifying broader schema items, such as relevant tables, is more effective than attempting to recognize specific columns, which may be ambiguous or difficult to map accurately. However, this strategy can limit the precision of the final query, especially when data granularity is essential. Overall, studies indicate that handling ambiguity and noise in data remains an open challenge, requiring improvements in data preprocessing techniques and model adaptation to manage imprecise or incomplete inputs.

Techniques for Overcoming Data Scarcity and Missing Information. Studies investigating NL-to-QL conversion face challenges related to data scarcity, missing, or unknown information and have adopted various strategies to overcome these limitations. Zhang et al. [91] proposed a hybrid data augmentation strategy that combines the use of LLMs with carefully designed rules to enrich the training set. This approach improves both the quality and quantity of available data, resulting in superior model performance. Additionally, Zhang implemented a parallel schema linking method using a Cross-Encoder model to retrieve schema elements relevant to the query, reducing noise and irrelevant content in the data input process, which facilitates the generation of more accurate SQL queries. Some studies [70, 71] also employed data augmentation techniques, such as rephrased datasets, to diversify the input text and enhance the model's robustness against linguistic variations. These strategies demonstrate that even with data limitations, it is possible to improve model performance, although challenges related to generalization and adaptation to new domains still persist.

Hallucination and Uncertainty of LLMs. The phenomenon of hallucination in LLMs refers to the situation that LLMs, when generating factual content, sometimes produce information that appears to be correct but contradicts actual facts [91]. The essence of this phenomenon lies in the LLMs' inability to maintain precise control over knowledge. Even the most powerful LLMs like GPT-4 may still encounter this issue. Agrahari et al. [2] hightlight that fine-tuning has proven to be very much effective in neural networks, however in LLM's it often induces a lot of hallucination after output is generated in smaller models, resulting in poor model's generation quality and overall poorly generated queries.

4.3 RQ3. How are the results evaluated?

RQ3 focuses on the metrics and benchmarks used to evaluate the results, considering how the quality of generated QL queries is defined and measured, the complexity standards of the commands produced, and examples that illustrate the sophistication of these queries.

Evaluating the performance of the NL-to-QL module becomes challenging when direct access to the real database is limited or restricted, like when we working on a bank database. It is essential to define suitable evaluation metrics that can measure the module's effectiveness without the need for executing queries on the actual database. Metrics such as logical form matching and semantic correctness can provide insights into the module's performance even when direct execution is not feasible.

Result quality assessment. Evaluating the performance of the NL-to-QL conversion module becomes challenging when direct access to the real database is limited or restricted, like when we working on a bank database [70]. Besides, the performance of LLMs is mainly influenced by the database prompt style, the exemplar selection strategy and prompt design [92]. Thus, to measure the results generated by LLMs, researchers have implemented various metrics grouped into three main categories, presented in Table 6.

Table 6: Metrics applied by the selected studies

Type	Metric	Advantages(A) / Disadvatages(D)
String-based match	EM	A: High efficiency / D: Cannot handle alias expressions
	CM	A: Can handle alias expressions / D: Need to be customized
	FM	A: Suitable for complexy queries /D: Insufficient precision
	LF	A: Avoids the need to run the QL / D: Need to be customized
Execution-based match	EX	A: Convenient, robust to alias expressions / D: Prone to false positives
	TS	A: Can handle semantically close expressions / D: Needs to be customized
Multi-turn approach	QM	A: Provide individual results / D: Needs to be customized
	IM	A: Provide overall results / D: Needs to be customized

The first group, String-based match, refers to textual correspondence between the generated query and the expected query (called gold data or oracle). This group consists of Exact Match Accuracy (EM), Component Match (CM), Fuzzy Match (FM), and Logical-Form Accuracy (LF).

- EM measures whether the QL query generated by the model is identical to the reference QL query (golden data), ignoring only minor, irrelevant variations (such as spaces or order of switchable clauses) [74].
- CM measures whether individual components of the generated QL query are correct, even if the query as a whole is not identical to the reference query [74].
- FM generates an approximate similarity index, whose score is based on string similarity. Although more flexible, this metric may ignore significant errors. [36]
- LF assesses the similarity of the generated query by comparing it with the ground truth query. It involves an exact string match to determine if the two queries are identical. [36]

Although string-based matching is not ideal for evaluating LLM performance in this context – as it may unfairly penalize models generating QL queries with stylistic differences from reference queries, even when the results are functionally equivalent – analyzing which components (fields, conditions, or values) deviate from the gold standard can pinpoint areas for improvement and guide targeted model enhancements [75].

The second group, Execution-based match, evaluates the degree of correctness of a QL query based on its execution results. This group consists of Execution Accuracy (EX), Test Suite Accuracy (TS).

- EX requires the execution result of the predicted QL to be correct. Since there may exist different QL queries that represent the same semantic, the EX metric is commonly more precise than the EM metric [21, 74, 92].
- TS serves as the official evaluation metric for Spider dataset. TS provides an upper-bound estimation of semantic accuracy by assessing the execution accuracy of predicted queries on a set of distilled randomly generated databases [74].

This metric also evaluates the execution result but requires the result to be correct under multiple database instances per database schema.

The third group, Manual evaluation, consists of human manually evaluations on generated queries against gold data. Despite being more costly, this approach increases the power of evaluation and can capture nuances and subtleties that may be lost in automated and limited processes. For multi-turn NL-to-QL datasets, we evaluate our approach with Question Match Accuracy (QM) and Interaction Match Accuracy (IM). Depending on what is being analyzed, these metrics may fit into any of the groups presented.

- The QM score is 1 if the predicted QL query for a single question is correct [21, 74].
- The IM score is 1 if all the predicted QL queries in the interaction are correct [92].

Competition and training platforms. There are several platforms that promote learning and competition between students and data engineers that aim to create solution to NL-to-QL conversion task. These platforms provide datasets, challenges, code examples, and a space for scientific publication of the best-placed ones. The platforms most cited in the selected papers were Spider 1.0, Spider 2.0, WikiSQL, and Kaggle.

Spider 1.0 [90] is a large-scale complex and cross-domain semantic parsing and NL-to-SQL dataset annotated by 11 Yale students. The goal of the Spider challenge is to develop natural language interfaces to cross-domain databases. It consists of 10,181 questions and 5,693 unique complex SQL queries on 200 databases with multiple tables covering 138 different domains. This platform provides two rankings related to the accuracy of methods in converting NL-to-QL. The first ranking is related to the Execution Accuracy (EX) metric. This list contains 33 methods, whose scores ranged from 53.5 to 91.2.

Spider 2.0 [43] is an evaluation framework comprising 632 real-world NL-to-SQL workflow problems derived from enterprise-level database use cases. The databases in Spider 2.0 are sourced from real data applications, often containing over 1,000 columns and stored in local or cloud database systems such as BigQuery and Snowflake. This challenge calls for models to interact with complex SQL workflow environments, process extremely long contexts, perform intricate reasoning, and generate multiple SQL queries with diverse operations, often exceeding 100 lines, which goes far beyond traditional NL-to-SQL challenges. This platform provides three ranking lists, all related to the Execution Accuracy (EX) metric. The first ranking shows results applied to the Spider 2.0-Snow dataset, with 18 methods with scores ranging from 0 to 31.26. The second ranking shows results applied to the Spider 2.0-lite dataset, with 15 methods with scores ranging from 0.73 to 30.5. The third ranking lists 8 methods with scores ranging from 2.1 to 17.01.

WikiSQL [87] consists of a corpus of 87,726 hand-annotated SQL query and natural language question pairs. These SQL queries are further split into training (61,297 examples), development (9,145 examples) and test sets (17,284 examples). It can be used for natural language inference tasks related to relational databases. Actually, it provides four benchmark with their results, scientific paper, and code. Kaggle [8] is a data science competition platform and online community for data scientists and machine learning practitioners under Google LLC. Kaggle enables users to find and publish datasets, explore and build models in a web-based data science environment, work with other data scientists and machine learning engineers, and enter competitions to solve data science challenges.

Other competition platforms for professionals and students involved in data science [65] are: DataDriven, CrowdAnalytix, Alibaba Tianchi, Innocentive, TunnedIT, CodaLab, Analytics Vidhya, CrowdAI, DataScienceChallenge, entre outras.

5 DISCUSSION

This systematic literature review reveals that NL-to-QL conversion task has evolved from rule-based approaches to neural network-driven techniques, with LLMs playing a transformative role. Despite advancements, challenges persist in query generalization, schema linking, and evaluation. We identified a growing trend in leveraging prompt engineering techniques and fine-tuning strategies to enhance model adaptability. Additionally, datasets and evaluation benchmarks such as Spider and WikiSQL remain pivotal but exhibit domain-specific limitations, highlighting the need for broader evaluation frameworks.

While traditional models focused on syntax and grammar rules, LLMs offer more flexibility through in-context learning. However, they are susceptible to hallucinations and require advanced prompt engineering techniques for optimization. Few-shot learning and chain-of-thought prompting have improved performance, yet challenges remain in ensuring consistency across different database schemas. The comparison highlights a shift towards hybrid models that integrate symbolic reasoning with deep learning for better generalization.

A major challenge in NL-to-QL conversion is handling schema variability across databases. Many models struggle with schema linking, leading to incorrect column or table associations. Additionally, LLMs are prone to hallucinations, generating syntactically correct but semantically incorrect queries. Data scarcity in domain-specific applications limits model generalization, and the lack of robust interpretability mechanisms hinders trust in automatically generated queries. These challenges underscore the need for improved context-aware models and methods that enhance user feedback integration.

Current evaluation metrics, such as exact match (EM) and execution accuracy (EX), do not fully capture semantic correctness. While execution-based metrics provide better real-world assessment, they can penalize correct queries with syntactic variations. Furthermore, the reliance on Spider and WikiSQL as benchmarks limits adaptability to industry-specific domains. Future evaluations should incorporate more diverse benchmarks, including multilingual and cross-domain datasets, to assess the robustness of NL-to-QL models comprehensively.

The findings suggest that LLMs have significant potential for real-world database querying, provided their limitations are addressed. The adoption of advanced prompting techniques and few-shot learning can improve model accuracy without extensive retraining. In practice, integrating LLMs into database management systems can simplify data retrieval for non-technical users, reducing dependency on QL command expertise. However, organizations must consider the risks of query misinterpretation and invest in mechanisms that validate model-generated queries before execution.

While SQL remains the most studied query language in academic research on NL-to-QL conversion, it no longer reflects the full spectrum of technologies adopted in modern data ecosystems. The increasing adoption of NoSQL databases—such as MongoDB, Cassandra, and Firebase—has led to the emergence of various non-relational query languages (e.g., MQL, CQL, FQL) and query interfaces like JQL, used in platforms such as Jira. According to the 2024 Stack Overflow Developer Survey [72], over 46% of respondents reported using NoSQL databases in their daily work, with MongoDB ranking among the top five most used databases worldwide.

Despite this trend, our findings show that only one of the 33 selected studies addressed GraphQL, and none explored JQL or CQL. This mismatch suggests that the state of the art is misaligned with the state of practice. Although some LLMs are already capable of generating commands in JQL [4, 19] and other NoSQL-related languages in commercial settings [34, 54], there is a lack of peer-reviewed research examining or evaluating such capabilities. Future studies should expand beyond SQL to ensure that research efforts align with the real-world diversity of database technologies and support broader adoption of NL-based interfaces in heterogeneous data environments.

6 THREATS TO VALIDITY AND LIMITATIONS

This study was designed to follow the guidelines of systematic literature reviews in software engineering, ensuring methodological rigor and transparency. However, several validity threats must be considered, following the classification proposed by Wohlin et al. [86].

Construct validity refers to the degree to which the extracted data and formulated research questions accurately reflect the phenomenon under investigation. Although the data extraction form was refined through pilot extractions and reviewed by multiple researchers, the interpretation of some elements—such as challenges or advantages—relies on the clarity and completeness of the primary studies. This may introduce bias due to inconsistent terminology or reporting across studies.

Internal validity relates to the reliability of the review process and whether the results can be attributed to the applied method rather than other factors. To mitigate this threat, dual screening and extraction were conducted independently, and disagreements were resolved through discussion with a third researcher. Nonetheless, the selection of studies was based solely on information retrievable via indexed databases and may have excluded relevant but non-indexed sources.

External validity concerns the generalizability of the findings. The inclusion criteria focused on English-language peer-reviewed publications from 2020 to 2024, which may limit the applicability of the results to other linguistic or cultural contexts. Additionally, the dominance of SQL in the reviewed studies (93%) restricts the extrapolation of findings to other query languages such as CQL or GraphQL, which remain underexplored.

Conclusion validity refers to the strength of the relationship between the data and the conclusions drawn. While thematic synthesis and content analysis were systematically applied, some subjectivity is inherent in qualitative synthesis. Furthermore, although several papers reported quantitative performance metrics, differences in experimental settings, datasets, and prompt designs reduce comparability, limiting the strength of cross-paper conclusions.

7 CONCLUSION

This systematic literature review mapped the state of the art in converting natural language to query languages (NL-to-QL), with a particular emphasis on the role of Large Language Models (LLMs). A total of 33 peer-reviewed studies published between 2020 and 2024 were analyzed, revealing the main methods adopted, key challenges faced, and evaluation strategies applied in this research domain.

Our findings indicate that LLMs have substantially advanced NL-to-QL conversion by enabling more flexible and accurate query generation through techniques such as prompt engineering and fine-tuning strategies. Despite these advances, critical issues persist, including difficulties with schema generalization, ambiguity resolution, and the risk of hallucinated outputs. Additionally, the dominance of SQL in existing research reveals dependency on specific benchmarks and a gap in addressing real-world demands, where NoSQL query languages are increasingly adopted.

From a practical perspective, this review highlights the potential of LLMs to democratize access to data by enabling non-technical users to interact with databases using natural language. However, to ensure reliability and safety, organizations must employ validation mechanisms and carefully design prompts tailored to the database context.

In terms of research implications, the study underscores the urgent need to expand the scope of NL-to-QL studies beyond SQL. Languages such as JQL, CQL, MQL, and GraphQL remain underexplored in academic research, despite their relevance in modern software ecosystems. Future work should focus on developing standardized benchmarks for these languages, conducting empirical evaluations in industrial settings, and proposing more robust models capable of adapting to diverse and dynamic database schemas.

In summary, while LLMs have reshaped the NL-to-QL landscape, realizing their full potential requires addressing current limitations and aligning research efforts with the practical needs of heterogeneous data environments.

ACKNOWLEDGMENTS

This paper is a result of the Research, Development & Innovation Project named COPILOTO at Sidia Institute of Science and Technology sponsored by Samsung Eletrônica da Amazônia Ltda., using resources under terms of Federal Law No. 8.387/1991, by having its disclosure and publicity in accordance with art. 39 of Decree No. 10.521/2020.

REFERENCES

[1] Willem Aerts, George Fletcher, and Daphne Miedema. 2024. A feasibility study on automated sql exercise generation with chatgpt-3.5. In *Proceedings of the 3rd International Workshop on Data Systems Education: Bridging education practice with education research.* 13–19.

[2] Ankit Agrahari, Puneet Kumar Ojha, Abhishek Gautam, and Parikshit Singh. 2024. SFT For Improved Text-to-SQL Translation. (2024).

[3] Aseem Arora, Shabbirhussain Bhaisaheb, Harshit Nigam, Manasi Patwardhan, Lovekesh Vig, and Gautam Shroff. 2023. Adapt and decompose: Efficient generalization of text-to-sql via domain adapted least-to-most prompting. *arXiv preprint arXiv:2308.02582* (2023).

[4] Atlassian Marketplace. 2025. JQL AI — Natural Language to JQL Query Assistant. https://marketplace.atlassian.com/apps/1237395/jql-ai. Accessed: 2025-07-17.

[5] Ayush Attawar, Shivam Vora, Parth Narechania, Vinaya Sawant, and Heli Vora. 2023. NLSQL: Generating and Executing SQL Queries via Natural Language Using Large Language Models. In *2023 International Conference on Advanced Computing Technologies and Applications (ICACTA).* IEEE, 1–6.

[6] Muhammad Shahzaib Baig, Azhar Imran, Aman Ullah Yasin, Abdul Haleem Butt, and Muhammad Imran Khan. 2022. Natural language to sql queries: A review. *International Journal of Innovations in Science Technology* 4 (2022), 147–162.

[7] Gourav Bathla, Pardeep Singh, and Rahul K Singh. 2021. AI-based Question Answering system for NoSQL standard query. (2021).

[8] Casper Solheim Bojer and Jens Peder Meldgaard. 2021. Kaggle forecasting competitions: An overlooked learning opportunity. *International Journal of Forecasting* 37, 2 (2021), 587–603.

[9] Ruichu Cai, Jinjie Yuan, Boyan Xu, and Zhifeng Hao. 2021. Sadga: Structure-aware dual graph aggregation network for text-to-sql. *Advances in Neural Information Processing Systems* 34 (2021), 7664–7676.

[10] Vanessa Câmara, Rayol Mendonca-Neto, André Silva, and Luiz Cordovil-Jr. 2023. DBVinci–towards the usage of GPT engine for processing SQL Queries. In *Proceedings of the 29th Brazilian Symposium on Multimedia and the Web.* 91–95.

[11] Shuaichen Chang and Eric Fosler-Lussier. 2023. Selective demonstrations for cross-domain text-to-SQL. *arXiv preprint arXiv:2310.06302* (2023).

[12] Xinyun Chen, Maxwell Lin, Nathanael Schärli, and Denny Zhou. 2023. Teaching large language models to self-debug. *arXiv preprint arXiv:2304.05128* (2023).

[13] DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *Computational Linguistics* 47, 2 (2021), 309–332.

[14] Edgar F Codd. 1970. A relational model of data for large shared data banks. *Commun. ACM* 13, 6 (1970), 377–387.

[15] Daniela S Cruzes and Tore Dyba. 2011. Recommended steps for thematic synthesis in software engineering. In *2011 international symposium on empirical software engineering and measurement.* IEEE, NJ, US, 275–284.

[16] Naihao Deng, Yulong Chen, and Yue Zhang. 2022. Recent advances in text-to-SQL: a survey of what we have and what we expect. *arXiv preprint arXiv:2208.10099* (2022).

[17] Xiang Deng, Ahmed Hassan Awadallah, Christopher Meek, Oleksandr Polozov, Huan Sun, and Matthew Richardson. 2020. Structure-grounded pretraining for text-to-SQL. *arXiv preprint arXiv:2010.12773* (2020).

[18] Anushka Deshpande, Dhruv Kothari, Akash Salvi, Prajakta Mane, and Vaishali Kolhe. 2022. Querylizer: An Interactive Platform for Database Design and Text to

SQL Conversion. In *2022 International Conference for Advancement in Technology (ICONAT)*. IEEE, 1–6.

[19] DocsBot AI. 2024. JQL Query Generator — Prompt for ChatGPT, Claude, Gemini. https://docsbot.ai/prompts/technical/jql-query-generator. Accessed: 2025-07-17.

[20] Steve Easterbrook, Janice Singer, Margaret-Anne Storey, and Daniela Damian. 2008. Selecting empirical methods for software engineering research. *Guide to advanced empirical software engineering* (2008), 285–311.

[21] Yuankai Fan, Tonghui Ren, Zhenying He, X Sean Wang, Ye Zhang, and Xingang Li. 2023. Gensql: A generative natural language interface to database systems. In *2023 IEEE 39th International Conference on Data Engineering (ICDE)*. IEEE, 3603–3606.

[22] Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R Woodward, John Drake, and Qiaofu Zhang. 2021. Natural SQL: Making SQL easier to infer from natural language specifications. *arXiv preprint arXiv:2109.05153* (2021).

[23] Dawei Gao, Haibin Wang, Yaliang Li, Xiuyu Sun, Yichen Qian, Bolin Ding, and Jingren Zhou. 2023. Text-to-sql empowered by large language models: A benchmark evaluation. *arXiv preprint arXiv:2308.15363* (2023).

[24] Youdi Gong, Guangzhen Liu, Yunzhi Xue, Rui Li, and Lingzhong Meng. 2023. A survey on dataset quality in machine learning. *Information and Software Technology* 162 (2023), 107268. https://doi.org/10.1016/j.infsof.2023.107268

[25] Chunxi Guo, Zhiliang Tian, Jintao Tang, Shasha Li, Zhihua Wen, Kaixuan Wang, and Ting Wang. 2023. Retrieval-augmented gpt-3.5-based text-to-sql framework with sample-aware prompting and dynamic revision chain. In *International Conference on Neural Information Processing*. Springer, 341–356.

[26] Chunxi Guo, Zhiliang Tian, Jintao Tang, Pancheng Wang, Zhihua Wen, Kang Yang, and Ting Wang. 2023. Prompting GPT-3.5 for Text-to-SQL with Desemanticization and Skeleton Retrieval. In *Pacific Rim International Conference on Artificial Intelligence*. Springer, 262–274.

[27] Jiaqi Guo, Zecheng Zhan, Yan Gao, Yan Xiao, Jian-Guang Lou, Ting Liu, and Dongmei Zhang. 2019. Towards complex text-to-sql in cross-domain database with intermediate representation. *arXiv preprint arXiv:1905.08205* (2019).

[28] Xinyi He, Mengyu Zhou, Xinrun Xu, Xiaojun Ma, Rui Ding, Lun Du, Yan Gao, Ran Jia, Xu Chen, Shi Han, et al. 2024. Text2analysis: A benchmark of table question answering with advanced data analysis and unclear queries. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 38. 18206–18215.

[29] Charles T Hemphill, John J Godfrey, and George R Doddington. 1990. The ATIS spoken language systems pilot corpus. In *Speech and Natural Language: Proceedings of a Workshop Held at Hidden Valley, Pennsylvania, June 24-27, 1990.*

[30] Hsiu-Fang Hsieh and Sarah E Shannon. 2005. Three approaches to qualitative content analysis. *Qualitative health research* 15, 9 (2005), 1277–1288.

[31] Edward J Hu, Yelong Shen, Phillip Wallis, Zeyuan Allen-Zhu, Yuanzhi Li, Shean Wang, Lu Wang, Weizhu Chen, et al. 2022. Lora: Low-rank adaptation of large language models. *ICLR* 1, 2 (2022), 3.

[32] Zhentao Hu, Wei Hou, and Xianxing Liu. 2024. Deep learning for named entity recognition: a survey. *Neural Computing and Applications* 36, 16 (2024), 8995–9022.

[33] Wonseok Hwang, Jinyeong Yim, Seunghyun Park, and Minjoon Seo. 2019. A comprehensive exploration on wikisql with table-aware word contextualization. *arXiv preprint arXiv:1902.01069* (2019).

[34] Dhruvisha Jaiswal. 2025. Building a Natural Language to MongoDB Query AI Agent. https://medium.com/@dhruvishajaiswal9/building-a-natural-language-to-mongodb-query-ai-agent-db7834f9435d. Accessed: 2025-07-17.

[35] Salomon Kabongo, Jennifer D'Souza, and Sören Auer. 2024. Exploring the latest LLMs for leaderboard extraction. *arXiv preprint arXiv:2406.04383* (2024).

[36] Ali Buğra Kanburoğlu and F Boray Tek. 2023. TUR2SQL: A cross-domain Turkish dataset for Text-to-SQL. In *2023 8th International Conference on Computer Science and Engineering (UBMK)*. IEEE, 206–211.

[37] Ali Buğra Kanburoğlu and Faik Boray Tek. 2024. Text-to-SQL: A methodical review of challenges and models. *Turkish Journal of Electrical Engineering and Computer Sciences* 32, 3 (2024), 403–419.

[38] George Katsogiannis-Meimarakis and Georgia Koutrika. 2021. A deep dive into deep learning approaches for text-to-sql systems. In *Proceedings of the 2021 International Conference on Management of Data*. 2846–2851.

[39] Imed Keraghel, Stanislas Morbieu, and Mohamed Nadif. 2024. A survey on recent advances in named entity recognition. *arXiv preprint arXiv:2401.10825* (2024).

[40] Barbara Kitchenham, Rialette Pretorius, David Budgen, O Pearl Brereton, Mark Turner, Mahmood Niazi, and Stephen Linkman. 2010. Systematic literature reviews in software engineering–a tertiary study. *Information and software technology* 52, 8 (2010), 792–805.

[41] Surjeet Kumar, Md Shamsher Alam, Zeeshan Khursheed, Shadan Bashar, and Neha Kalam. 2024. Enhancing Relational Database Interaction through Open AI and Stanford Core NLP-Based on Natural Language Interface. In *2024 5th International Conference on Recent Trends in Computer Science and Technology (ICRTCST)*. IEEE, 589–602.

[42] J. R. Landis and G. G. Koch. 1977. The Measurement of Observer Agreement for Categorical Data. *Biometrics* 33, 1 (1977), 159–174. https://doi.org/10.2307/2529310

[43] Fangyu Lei, Jixuan Chen, Yuxiao Ye, Ruisheng Cao, Dongchan Shin, Hongjin Su, Zhaoqing Suo, Hongcheng Gao, Wenjing Hu, Pengcheng Yin, et al. 2024. Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows. *arXiv preprint arXiv:2411.07763* (2024).

[44] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, et al. 2024. 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 (2024).

[45] Jinqing Lian, Xinyi Liu, Yingxia Shao, Yang Dong, Ming Wang, Zhang Wei, Tianqi Wan, Ming Dong, and Hailin Yan. 2024. ChatBI: Towards natural language to complex business intelligence SQL. *arXiv preprint arXiv:2405.00527* (2024).

[46] Jinfeng Lin, Yalin Liu, Jin Guo, Jane Cleland-Huang, William Goss, Wenchuang Liu, Sugandha Lohar, Natawut Monaikul, and Alexander Rasin. 2017. Tiqi: A natural language interface for querying software project data. In *2017 32nd IEEE/ACM International Conference on Automated Software Engineering (ASE)*. IEEE, 973–977.

[47] Aixin Liu, Bei Feng, Bing Xue, Bingxuan Wang, Bochao Wu, Chengda Lu, Chenggang Zhao, Chengqi Deng, Chenyu Zhang, Chong Ruan, et al. 2024. Deepseek-v3 technical report. *arXiv preprint arXiv:2412.19437* (2024).

[48] Pan Liu, Yanming Guo, Fenglei Wang, and Guohui Li. 2022. Chinese named entity recognition: The state of the art. *Neurocomputing* 473 (2022), 37–53.

[49] Shicheng Liu, Jialiang Xu, Wesley Tjangnaka, Sina J Semnani, Chen Jie Yu, and Monica S Lam. 2023. SUQL: Conversational Search over Structured and Unstructured Data with Large Language Models. *arXiv preprint arXiv:2311.09818* (2023).

[50] Qin Lyu, Kaushik Chakrabarti, Shobhit Hathi, Souvik Kundu, Jianwen Zhang, and Zheng Chen. 2020. Hybrid ranking network for text-to-sql. *arXiv preprint arXiv:2008.04759* (2020).

[51] Yuchi Ma, Zhou Zhang, Hsiuhan Lexie Yang, and Zhengwei Yang. 2021. An adaptive adversarial domain adaptation approach for corn yield prediction. *Computers and Electronics in Agriculture* 187 (2021), 106314.

[52] Alaa Marshan, Anwar Nais Almutairi, Athina Ioannou, David Bell, Asmat Monaghan, and Mahir Arzoky. 2024. MedT5SQL: a transformers-based large language model for text-to-SQL conversion in the healthcare domain. *Frontiers in Big Data* 7 (2024), 1371680.

[53] Y Mellah, A Rhouati, EH Ettifouri, T Bouchentouf, and MG Belkasmi. 2021. SQL generation from natural language: A sequence-to-sequence model powered by the transformers architecture and association rules. *Journal of Computer Science* 17, 5 (2021), 480–489.

[54] MongoDB, Inc. 2025. Query with Natural Language in MongoDB Compass. https://www.mongodb.com/docs/compass/current/query-with-natural-language/prompt-natural-language-query/. Accessed: 2025-07-17.

[55] Linyong Nan, Yilun Zhao, Weijin Zou, Narutatsu Ri, Jaesung Tae, Ellen Zhang, Arman Cohan, and Dragomir Radev. 2023. Enhancing text-to-SQL capabilities of large language models: A study on prompt design strategies. In *Findings of the Association for Computational Linguistics: EMNLP 2023*. 14935–14956.

[56] Eduardo R Nascimento, Grettel M Garcıa, Lucas Feijó, Wendy Z Victorio, Yenier T Izquierdo, Aiko R de Oliveira, Gustavo MC Coelho, Melissa Lemos, Robinson LS Garcia, Luiz AP Paes Leme, et al. 2024. Text-to-SQL Meets the Real-World. In *26th Int. Conf. on Enterprise Info. Sys.*

[57] Eduardo R Nascimento, Yenier T Izquierdo, Grettel M Garcıa, Gustavo MC Coelho, Lucas Feijó, Melissa Lemos, Luiz AP Paes Leme, and Marco A Casanova. 2024. My Database User Is a Large Language Model. In *26th Int. Conf. on Enterprise Info. Sys.*

[58] Pin Ni, Ramin Okhrati, Steven Guan, and Victor Chang. 2024. Knowledge graph and deep learning-based text-to-GraphQL model for intelligent medical consultation chatbot. *Information Systems Frontiers* 26, 1 (2024), 137–156.

[59] Arkil Patel, Satwik Bhattamishra, Siva Reddy, and Dzmitry Bahdanau. 2023. Magnifico: Evaluating the in-context learning ability of large language models to generalize to novel interpretations. *arXiv preprint arXiv:2310.11634* (2023).

[60] Yuriy Perezhohin. 2023. *A Neural Rule-Based Model for Database Exploration: The Combination of Rule-Based Methods With Pre-Trained Embedding Models for Text to SQL Task*. Master's thesis. Universidade NOVA de Lisboa (Portugal).

[61] Putsadee Pornphol and Suphamit Chittayasothorn. 2024. Using LLM Artificial Intelligence Systems as Complex SQL Programming Assistants. In *2024 12th International Conference on Information and Education Technology (ICIET)*. IEEE, 477–481.

[62] Mohammadreza Pourreza and Davood Rafiei. 2024. Din-sql: Decomposed in-context learning of text-to-sql with self-correction. *Advances in Neural Information Processing Systems* 36 (2024).

[63] Bowen Qin, Binyuan Hui, Lihan Wang, Min Yang, Jinyang Li, Binhua Li, Ruiying Geng, Rongyu Cao, Jian Sun, Luo Si, et al. 2022. A survey on text-to-sql parsing: Concepts, methods, and future directions. *arXiv preprint arXiv:2208.13629* (2022).

[64] Federico Ranaldi, Elena Sofia Ruzzetti, Leonardo Ranaldi, Davide Venditti, Cristina Giannone, Andrea Favalli, Raniero Romagnoli, Fabio Massimo Zanzotto, et al. 2023. Prompting LLMs in Italian Language for Text-to-SQL Translation.. In *CLiC-it*.

[65] Marlesson Santana. 2019. *As melhores plataformas de Competição para Cientistas de Dados*. https://medium.com/data-hackers/plataformas-de-competi%C3%A7%C3%A3o-para-cientistas-de-dados-a26f86fdbda3

[66] Shiv Shankar, Vihari Piratla, Soumen Chakrabarti, Siddhartha Chaudhuri, Preethi Jyothi, and Sunita Sarawagi. 2018. Generalizing across domains via cross-gradient training. *arXiv preprint arXiv:1804.10745* (2018).

[67] Ankita Sharma, Xuanmao Li, Hong Guan, Guoxin Sun, Liang Zhang, Lanjun Wang, Kesheng Wu, Lei Cao, Erkang Zhu, Alexander Sim, et al. 2023. Automatic data transformation using large language model-an experimental study on building energy data. In *2023 IEEE International Conference on Big Data (BigData)*. IEEE, 1824–1834.

[68] Hao Shen, Ran Shen, Gang Sun, Yiling Li, Yifan Wang, and Pengcheng Zhang. 2023. Sequential Feature Augmentation for Robust Text-to-SQL. In *2023 International Conference on Algorithms, Computing and Data Processing (ACDP)*. IEEE, 217–223.

[69] Peng Shi, Rui Zhang, He Bai, and Jimmy Lin. 2022. Xricl: Cross-lingual retrieval-augmented in-context learning for cross-lingual text-to-sql semantic parsing. *arXiv preprint arXiv:2210.13693* (2022).

[70] Yewei Song, Saad Ezzini, Xunzhu Tang, Cedric Lothritz, Jacques Klein, Tegawendé Bissyandé, Andrey Boytsov, Ulrick Ble, and Anne Goujon. 2024. Enhancing Text-to-SQL Translation for Financial System Design. In *Proceedings of the 46th International Conference on Software Engineering: Software Engineering in Practice*. 252–262.

[71] Yuanfeng Song, Raymond Chi-Wing Wong, and Xuefang Zhao. 2024. Speech-to-SQL: toward speech-driven SQL query generation from natural language question. *The VLDB Journal* (2024), 1–23.

[72] Stack Overflow. 2024. 2024 Stack Overflow Developer Survey. https://survey.stackoverflow.co/2024/#technology-databases. Accessed: 2025-07-17.

[73] Alane Suhr, Ming-Wei Chang, Peter Shaw, and Kenton Lee. 2020. Exploring unexplored generalization challenges for cross-database semantic parsing. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 8372–8388.

[74] Ruoxi Sun, Sercan Ö Arik, Rajarishi Sinha, Hootan Nakhost, Hanjun Dai, Pengcheng Yin, and Tomas Pfister. 2023. Sqlprompt: In-context text-to-sql with minimal labeled data. *arXiv preprint arXiv:2311.02883* (2023).

[75] Shuo Sun, Yuchen Zhang, Jiahuan Yan, Yuze Gao, Donovan Ong, Bin Chen, and Jian Su. 2023. Battle of the Large Language Models: Dolly vs LLaMA vs Vicuna vs Guanaco vs Bard vs ChatGPT–A Text-to-SQL Parsing Comparison. *arXiv preprint arXiv:2310.10190* (2023).

[76] Chang-You Tai, Ziru Chen, Tianshu Zhang, Xiang Deng, and Huan Sun. 2023. Exploring chain-of-thought style prompting for text-to-sql. *arXiv preprint arXiv:2305.14215* (2023).

[77] Toni Taipalus. 2024. Database management system performance comparisons: A systematic literature review. *Journal of Systems and Software* 208 (2024), 111872. https://doi.org/10.1016/j.jss.2023.111872

[78] Zhao Tan, Xiping Liu, Qing Shu, Xi Li, Changxuan Wan, Dexi Liu, Qizhi Wan, and Guoqiong Liao. 2024. Enhancing Text-to-SQL Capabilities of Large Language Models through Tailored Promptings. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*. 6091–6109.

[79] Jingwei Tang, Guodao Sun, Jiahui Chen, Gefei Zhang, Baofeng Chang, Haixia Wang, and Ronghua Liang. 2024. MAVIDSQL: A Model-Agnostic Visualization for Interpretation and Diagnosis of Text-to-SQL Tasks. *IEEE Transactions on Cognitive and Developmental Systems* (2024).

[80] Alessandro Tola. 2024. *Towards user-friendly nosql: A synthetic dataset approach and large language models for natural language query translation*. Ph.D. Dissertation. Politecnico di Torino.

[81] Immanuel Trummer. 2023. Demonstrating gpt-db: Generating query-specific and customizable code for sql processing with gpt-4. *Proceedings of the VLDB Endowment* 16, 12 (2023), 4098–4101.

[82] Shao V Tsiu, Mfanelo Ngobeni, Lesley Mathabela, and Bonginkosi Thango. 2025. Applications and competitive advantages of data mining and business intelligence in SMEs performance: A systematic review. *Businesses* 5, 2 (2025), 22.

[83] Haijing Tu. 2024. Cassandra vs. mongodb: A systematic review of two nosql data stores in their industry uses. In *2024 IEEE 7th International Conference on Big Data and Artificial Intelligence (BDAI)*. IEEE, 81–86.

[84] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems* 30 (2017).

[85] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7567–7578.

[86] Claes Wohlin, Per Runeson, Martin Höst, Magnus C. Ohlsson, Björn Regnell, and Anders Wesslén. 2012. *Experimentation in Software Engineering*. Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-642-29044-2

[87] Kun Xu, Lingfei Wu, Zhiguo Wang, Yansong Feng, and Vadim Sheinin. 2018. Sql-to-text generation with graph-to-sequence model. *arXiv preprint arXiv:1809.05255* (2018).

[88] Xiaojun Xu, Chang Liu, and Dawn Song. 2017. Sqlnet: Generating structured queries from natural language without reinforcement learning. *arXiv preprint arXiv:1711.04436* (2017).

[89] Tao Yu, Zifan Li, Zilin Zhang, Rui Zhang, and Dragomir Radev. 2018. Typesql: Knowledge-based type-aware neural text-to-sql generation. *arXiv preprint arXiv:1804.09769* (2018).

[90] Tao Yu, Rui Zhang, Kai Yang, Michihiro Yasunaga, Dongxu Wang, Zifan Li, James Ma, Irene Li, Qingning Yao, Shanelle Roman, 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).

[91] Chao Zhang, Yuren Mao, Yijiang Fan, Yu Mi, Yunjun Gao, Lu Chen, Dongfang Lou, and Jinshu Lin. 2024. FinSQL: Model-Agnostic LLMs-based Text-to-SQL Framework for Financial Analysis. *arXiv preprint arXiv:2401.10506* (2024).

[92] Hanchong Zhang, Ruisheng Cao, Lu Chen, Hongshen Xu, and Kai Yu. 2023. ACT-SQL: In-Context Learning for Text-to-SQL with Automatically-Generated Chain-of-Thought. *arXiv preprint arXiv:2310.17342* (2023).

[93] Tianyi Zhang, Tao Yu, Tatsunori Hashimoto, Mike Lewis, Wen-tau Yih, Daniel Fried, and Sida Wang. 2023. Coder reviewer reranking for code generation. In *International Conference on Machine Learning*. PMLR, 41832–41846.

[94] Weixu Zhang, Yifei Wang, Yuanfeng Song, Victor Junqiu Wei, Yuxing Tian, Yiyan Qi, Jonathan H Chan, Raymond Chi-Wing Wong, and Haiqin Yang. 2024. Natural language interfaces for tabular data querying and visualization: A survey. *IEEE Transactions on Knowledge and Data Engineering* (2024).

[95] Xiaoyu Zhang, Fengjing Yin, Guojie Ma, Bin Ge, and Weidong Xiao. 2020. M-SQL: Multi-task representation learning for single-table Text2sql generation. *IEEE Access* 8 (2020), 43156–43167.

[96] Yusen Zhang, Jun Wang, Zhiguo Wang, and Rui Zhang. 2023. XSemPLR: Cross-lingual semantic parsing in multiple natural languages and meaning representations. *arXiv preprint arXiv:2306.04085* (2023).