Brandenburg University of Applied Sciences

IT Security
Computerscience
Prof. Dr. Oleg Lobachev
MSc. Florian Eich

Reliable Natural Language Interfaces using LLMs, Self-Correction and Incremental Schema Analysis

Bachelor Thesis

Summer semester 2025
May 17, 2025

Mara Schulke – Matr-Nr. 20215853

Abstract

This thesis explores the integration of large language models (LLMs) into PostgreSQL database systems in order to make the database accessible via natural language instead of the postgres SQL dialect. The research focuses on implementation strategies, performance optimization, and practical applications of this concept.

Contents

List of Figures

List of Abbreviations

GPT	Generative Pretrained Transformer
SQL	Structured Query Language
API	Application Programming Interface
LLM	Large Language Model
DBMS	Database Management System
NL2SQL	Natural Language to SQL

1 Introduction

1.1 Problem Statement and Motivation

Database systems represent a backbone of modern computer science, allowing for rapid advancements whilst shielding us from the problem categories that come along with managing and querying large amounts of, usually structured, data efficiently. However, most Database Management Systems (DBMS) have traditionally required specialized knowledge, usually of the Structured Query Language (SQL), in order to become useable. Whilst this barrier may be percieved differently across diverse usergroups it represents a fundamental misalignment between end-user goals (e.g. analysts, researchers, domain experts etc.) and the underlying DBMS, thus often requiring software engineering efforts in order to reduce this friction.

This barrier is the reason entire classes of software projects exists (for example, admin / support panels), data analytics tools etc. which therefore introduce significant churn and delay between the implementation of a database system and reaching the desired end user impact. Often these projects span multiple years, require costly staffing and yield little to no novel technical value.

Emerging technologies such as Large Language Models (LLMs) have proven themselves as a sensible tool for bridging fuzzy user provided input into discrete, machine readable formats. Prominent models in this field have demostrated outstanding capabilities that enable computer scientists to tackle new problem classes, that used to be challenging / yielded unsatisfying results with logical programming approaches.

This thesis is exploring ways to overcome the above outlined barrier using natural language queries, so that domain experts, business owners, support staff etc. are able to seamlessly interact with their data, essentially eliminating the requirement of learning SQL (and its pitfalls). By translating natural language to SQL using Large Language Models this translation becomes very robust (e.g. against different kinds of phrasing) and enables novel applications in how businesses, researchers and professionals interact with their data — it represents a fundamental shift (ie. moving away from SQL) towards a more inclusive and data driven world.

1.2 Objectives of the Thesis

This thesis aims to address the aforementioned challanges when it comes to database accessibility. The following objectives are the core research area of this thesis:

1. Develop a database extension that can translate natural language queries into semantically accurate SQL queries using Large Language Models.

2. To evaluate the effectiveness and feasibility of different Models aswell as prompt engineering techniques in order to improve the performance of the system.

3. Identify and address issues when it comes to handling amibguous, complex and domain specific user input.

4. Benchmark the performance of the implementation against common natural language to SQL (NL2SQL) benchmarks.

5. Idenitfy potential use cases for real world scenarios that could deliver a noticable upsides to users.

6. Analyze the short commings and limitations of this approach and propose potential solutions to overcome them.

1.3 Research Questions

RQ1 — Are natural language database interfaces feasible for real world application?

The primary research questions when it comes to natural language database interfaces evolve around their semantic accuracy and reliability, therefore questioning their feasibility for real world usage. LLMs have notoriously been known for their ability to hallucinate / produce false, but promising outputs. This behaviour can be especially dangerous when opting for data driven decisions that rely on false data due to a mistranslation from natural language to SQL. LLMs could cause hard to understand and debug behaviour, like false computation of distributions when the intermediate format is not being shown to the user. This thesis tries to determine whether such hallucinations could be reasonably prevented and whether the associated performance and hardware requirements are suitable for a real world deployment, outside of research situations.

Specifically the two big underlying questions are:

1. Is the semantic accuracy of natural language database interfaces high enough to yield a noticable benefit to users?
2. Is it possible to run such an interface on reasonable, mass available hardware (e.g. excluding high end research GPUs).

RQ2 — What approaches are most effective in resolving ambiguity when translating natural language queries into SQL?

To provide semantically correct results ambiguity in the user-provided natural language queries must be adequately addressed. This thesis investigates various approaches to ambiguity management and resolution. Natural language queries can demonstrate ambiguity even at low levels of complexity — e.g. there are two different types of "sales" in a database schema, and the user asks to retireve "all sales".

Such situations present the second major challenge associated with the practical implementation of natural language database interfaces. The success of this concept will significantly depend on whether suitable designs and mitigation techniques can be implemented without creating problems with regards to the aforementioned performance and hardware requirements. The research focus lies on both preventative measures through optimized pre-processing stages and prompt engineering techniques as well as reactive strategies that post process LLM output, either on the basis of further user input or context inference.

RQ3 — Which strategies are increasing semantic accuracy of queries?

In order to enhance the semantic accuracy a series of improvements may be applied to the pipeline. Potential optimizations include supplying (parts of) the schema during LLM prompting, implementation of interactive contextual reasoning through a conversational interface which would allow for user refinement, the implementation of a robust SQL parsing and validation mechanism and a hybrid approach partly relying on traditional NLP preprocessing techniques. This research will quantify semantic accuracy using popular NL2SQL benchmarks and empirically evaluate the impact each approach has on the benchmark performance. Furthermore this research will take a look at the optimal combination of the aforementioned solutions in order to develop a system that strikes the right balance between accuracy and performance.

1.4 Structure of the Thesis

This thesis is following a research and development methodology in order to implement a natural language interface for databases, in particular postgres is used.

1. **Literature Review** — An analysis of the existing research in the fields of natural language interfaces (NLI) for databases, GPU integration for acceleration of database operations, and LLM/AI Model integration within database systems. This phase establishes the theoretical foundation for this research and identifies current state-of-the-art approaches, their benefits and shortcomings.

2. **Decomposition & Requirements** — Decomposing the problem statement into its fundamentals and deriving system requirements for the design phase from it. The goal of this section is to arrive at a list of functional and non-functional requirements that must be taken into account and fulfilled by the design and implementation phases respectively.

3. **System Design** — Design of a system architecture that can utilize GPU acceleration for LLM integration from within postgres. The primary goals of the system design phase are to arrive at an architecture that yields low latency natural language processing, schema-aware SQL query generation, ambiguity detection and resolution whilst maintaining a high semantic accuracy.

4. **Implementation** — The implementation of a PostgreSQL extension according to the above system design that relies on `rust` and `pgrx`. This extension will provide a GPU accelerated framework for executing LLMs, implement a natural language to query generation pipeline that relies on the SQL schema and create database functions and operators for both query generation and execution.

5. **Evaluation and Benchmarking** — An assesment framework and benchmark that introspects the implementations performance in multiple dimensions. Namely the most relevant dimensions for this thesis are:

   (a) Semantic Accuracy — Measuring the overall accuracy of results delivered for a given natural language input.

   (b) Ambiguity Resolution Capabilities — How well the system performs when confronted with ambiguous natural language input and database schemas.

   (c) Performance Metrics — Measuring the latency, throughput and resource utilization of the implementation.

6. **Discussion** — Analysis and interpretation of the evaluation phase results against the research goals of this thesis. Evaluating the performance and accuracy results recorded during the benchmarks against the question whether real world deployments of NILs are feasible. Furthermore the effectiveness of ambiguity resolution capabilities and semantic accuracy enhancement strategies are showing a statistically significant effect.

7. **Summary and Outlook** — Summarizes the contributions, addresses limitations of this thesis and the implementation, and proposes directions for future research alongside possible applications. Primary future research topics include advanced GPU optimization techniques (e.g. further quantization), accuracy and performance impact of model fine tuning, techniques, scalability of such a system in enterprise scenarios and the evaluation of security and privacy considerations (e.g. managing access control).

2 Literature Review

In this section a comprehensive literature review is performed to asses the research landscape on NL2SQL (sometimes also referred to as Text-to-SQL or T2SQL) and NLIDBs. From the time their development accelerated in the late 1990s and early 2000s (Androutsopoulos, Ritchie, & Thanisch, 1995; Popescu, Etzioni, & Kautz, 2003; Tang & Mooney, 2001; Zelle & Mooney, 1996) until now, observing multiple larger paradigm shifts happening over time (Deng et al., 2020; F. Li & Jagadish, 2014; Yaghmazadeh, Wang, Dillig, & Dillig, 2017; Yu et al., 2020; Zhong, Xiong, & Socher, 2017). In particular this research focuses on the recent advancements when it comes to language models and how they can be harnessed for effective NL2SQL systems (D. Gao et al., 2023; Lei et al., 2025; J. Li, Hui, Qu, et al., 2023; Rahaman, Zheng, Milani, Chiang, & Pottinger, 2024; Rajkumar, Li, & Bahdanau, 2022; B. Zhang et al., 2024).

This literature review is covering the foundational concepts, challenges, key advancements and research gaps associated with using natural language instead of SQL. It lays the foundation for this thesis and helps to set the research questions introduced in the previous chapter in context.

2.1 Foundations of Natural Language Interfaces to Databases

Earlier papers in the research landscape on Natural Language Database Interfaces (NLIDBs) date over half a century back, into the early 1970s. Two decades after the first major research systems where developed in this domain, Androutsopoulos, Ritchie, and Thanisch have published an introduction and an overview over NLIDBs where an overview of state-of-the-art approaches were provided. (Androutsopoulos et al., 1995) Their work outlined multiple key issues and challenges associated with NLIDBs, and compared them against existing / competing solutions like formal query languages, form-based interfaces and graphical interfaces. These challenges (like unobvious limits, linguistic ambiguities, semantic inaccuracy, tedious configuration etc.) have shaped this field of research and are still considered relevant metrics today.

Early NLIDBs primarily relied on traditional natural language processing (NLP) techniques in order to achieve natural language understanding capabilities. With Chill an inductive logic programming (ILP) approach was first introduced for NL2SQL systems, marking one of the key events when it comes to machine learning usage. (Zelle & Mooney, 1996) In 2001 Tang and Mooney have extended the approach of ILP parsing for natural language database queries with multi clause construction, yielding promising results in the field of NLIDBs. (Tang & Mooney, 2001)

Building on the systematic overview of Androutsopoulos, Ritchie, and Thanisch and the first machine learning approaches from Zelle and Mooney aswell as Tang and Mooney, Popescu et al. have proposed a novel approach for implementing NLIDBs and outperformed at the time state-of-the-art solutions from Zelle and Mooney (1996) Tang and Mooney (2001) — achieving 80% semantic accuracy. (Popescu et al., 2003) The novelty of the Percise system lies in its natural language processing approach, specifically its lexical mapping strategy, allowing Percise to identify questions it can, and can't answer (introducing the concept of *semantically tractable questions*) which therefore results in a better and interactive end user experience. Their experiements also showed that this approach is *transferrable* and *unbiased* — it is possible to feed in new, unknown questions into the system and maintain performance characteristics, whereas it was shown that Zelle and Mooney (1996) were suffering from a distribution drift of the questions asked. (Popescu et al., 2003)

The theoretical foundations and research questions highlighted by the aforementioned works, shaped the research field and highlighted the following, ongoing research:

1. The trade-off characteristics derived from choosing a machine learning vs. traditional NLP approach (e.g. Chill versus Percise). E.g. coverage versus correctness. (Popescu et al., 2003; Zelle & Mooney, 1996)
2. The linguistic challenges associated with bringing NLIDBs into use (e.g. semantic inaccuracy, linguistic ambiguity, unclear language coverage etc.) (Androutsopoulos et al., 1995)
3. The value of systems and approaches which double down on reliability and semantic accuracy rather than giving promising but incorrect answers. (Androutsopoulos et al., 1995; Popescu et al., 2003)

Fundamentally this highlights the tension and mismatch between the characteristics of natural language, which is able to be ambiguous, *semantically untractable* or able to be incomplete in meaning and formal languages like SQL which always have on deterministic and *semantically tractable* meaning they convey in each statement. As Schneiderman and Norman have pointed out according to Popescu, Etzioni, and Kautz, users are "unwilling to trade reliable and predictable user interfaces for intelligent but unreliable ones" which induces performance expectations on NLIDB implementations to be highly certain about the questions it can, and can't answer, whilst maintaing as high as possible natural language coverage. (Popescu et al., 2003)

2.2 Traditional NL2SQL Approaches

Prior to the wide-spread dominance of machine learning approaches for natural language processing a variety of traditional, rather discrtete approaches have been explored in the field of NL2SQL / NLIDBs. These logical programming approaches have laid the foundations for transitioning towards the application of machine learning techniques for NL2SQL.

2.2.1 Rule-based and Grammar-based Systems

Foundational research of NL2SQL system mostly focused around applying rule engines that were tedious to set up and expensive to maintain / transfer across database systems. These rule engines mostly relied on the systematic identification of linguistic patterns / were trying to template SQL from information that was derived from processing the natural language query. (Codd, 1974; Hendrix, Sacerdoti, Sagalowicz, & Slocum, 1978; Woods, Kaplan, & Nash-Webber, 1972) These approaches mostly tried to formalize natural language queries into formal grammers which could then be deterministically mapped into a valid SQL query. (Woods et al., 1972) These approaches have strong downsides when it comes to the variety of natural language constructs they can process, aswell as runtime adoption of new / unknown databases, query constructs etc. A potential upside of this class of NL2SQL systems is that they can confidently and reproducably identify questions they can, and can't answer — thus leading to very reliable and predictable user interfaces.

2.2.2 Semantic Parsing using String-Kernels

A significiant milestone in parsing techniques of natural language queries was reached by Kate and Mooney in 2006. The introduction of string kernels for semantic parsing represented a novel achievement, when it comes to fusing logical programming approaches using a formal grammer like LSNLIS developed by Woods et al. (1972) and learning / training approaches to understand unseen language patterns / unknown natural language query structures. This allowed for more flexible pattern recognition when compared to traditional rule-based systems.

The core innovative characteristic of this approach lies in its capability to understand similarities between natural language expressions based on subsequence patterns rather than relying on exact matches. This made Krisp, the research NLIDB system developed by Kate and Mooney (2006) much more robust to language variations in phrasing and noise (e.g. spelling mistakes) in the input. As the Kate and Mooney demonstrated through experiments on real-world datasets, this approach compared favorably to existing systems of the time like Chill, especially in handling noisy inputs — a frequent challenge rigid rule-based systems faced in real world scenarios (Kate & Mooney, 2006; Zelle & Mooney, 1996).

2.2.3 Graph Matching Methods

Reddy, Lapata, and Steedman (2014) brought together several research threads and reapplied emerging graph matching research models to natural language processing, specifically to natural language queries. Graph matching was applied once the natural language query was parsed using a Combinatory Categorial Grammar (CCG) approach into a semantic graph which denotes the relationship between semantic entities in it. This graph could then be matched against the actual graph derived from the database, since they share topological traits that can be used for matching (Reddy et al., 2014). This approach allowed to apply querying systems without having any question-answer pairs or manual annotations for training the system, which implies easier scalability / transferability across domains, since the system does not require any additional tweaks.

Even though this approach was novel and showed improved the performance over existing state-of-the-art approaches, it was showing that graph matching quickly reaches its limitations. This approach relied heavily on the CCG parser's accuracy, with parsing errors accounting for 10-25% (depending on the dataset) of system failures (Reddy et al., 2014). Furthermore it struggled with both ambiguous language constructs and potential mismatches between natural language representation of relationships and database layouts — more complicated database designs, which may not match the users intuitive understanding resulted in a different topology and hence could not be matched (Reddy et al., 2014, p. 387).

2.2.4 Interactive Systems

In 2014 F. Li and Jagadish identified that perfect translation of natural language into SQL was challenging due to natural language not being made for query expressions as it heavily relies on contextual information and clarifying questions in order to disambiguate conversations (F. Li & Jagadish, 2014). These learnings relate to early prior art from Montgomery and Codd which also made this observation — "natural language is not a natural query language." (Montgomery, 1972). The solution introduced by NaLIR further emphasized how important an interactive, conversational usage model is, when offering a natural language interface (F. Li & Jagadish, 2014).

NaLIR could accept logically complex English language sentences as input and translate them into SQL queries with various complexities, including aggregation, nesting, and different types of joins etc. The key innovative characteristic of NaLIR lies in its interactive communication mechanism (much like Rendezvous) that could detect potential misinterpretations and engage users to resolve ambiguities present in their natural language query without forcing them to entirely rephrase their query F. Li and Jagadish (2014). This approach, while showing awareness for its limitations (with regards to entirely automating / deriving SQL generation from potentially ambiguous or faulty user input) showed that it was possible to overcome these limtations through choosing the right interaction model — "In our system, we generate multiple possible interpretations for a natural language query and translate all of them in natural language for the user to choose from" —, rather than optimizing the generation part of the system F. Li and Jagadish (2014).

2.2.5 Query Synthesis

Yaghmazadeh et al. (2017) introduced SQLizer, which synthesizes SQL queries from natural language (Yaghmazadeh et al., 2017). This paper presents a novel approach when it comes to NL2SQL as it is merging prevalent semantic parsing techniques (outlined above) with an program synthesis (or query synthesis) approach. SQLizer makes use of a three stage processing model for natural language models: first generating a sketch of the query using semantic parsing, then using type-directed synthesis to complete the sketch and finally using automated repair, if required.

Yaghmazadeh, Wang, Dillig, and Dillig show that alternating between repairing and synthesis yields results that beat state-of-the-art NL2SQL approaches like NaLIR. SQLizer is fully automated and database-agnostic, requiring no knowledge of the underlying schema. The authors evaluated SQLizer on 455 queries across three databases, where it ranked the correct query in the top 5 results for roughly 90% of the queries. This represents a significant improvement over NaLIR (F. Li & Jagadish, 2014), the previous state-of-the-art system (Yaghmazadeh et al., 2017).

Potential short commings of this approch include queries which yield empty results, dealing with language variations as SQLizer is still using semantic parsing, and domain-specific terminology, all while still requiring users to select from multiple query options which reduces the overall usability of the system (Yaghmazadeh et al., 2017, p.22-23).

2.2.6 Limitations of Traditional Approaches to NL2SQL

Despite being innovative and achieving state-of-the-art results, many of the above outlined approaches face severe challenges when moving outside of an research environment. Many of these systems performed comparatively good on research benchmarks that were often composed of controlled question types and limited data variety. Ultimatively no standard benchmark existed for NL2SQL in this era, hence comparing different NL2SQL systems against each other is a problem on its own. Despite not having a standard benchmark that all approaches could be unifiably evaluated against, several fundamental challenges emerged / remained with these approaches:

1. **Limited linguistic coverage** — Prevalent rule-based and semantic-parsing based systems were only able to process the a small subset of the natural language they were programmed for. This severely limited their ability to handle different phrasings of the same end-user goal (Hendrix et al., 1978; Kate & Mooney, 2006; Montgomery, 1972; Woods et al., 1972).

2. **Transferability** — Traditional approaches typically required extensive manual configuration or at least a training phase / adaption for each database they were deployed for, hindering cross domain usage through being expensive and time-consuming to adapt (Androutsopoulos et al., 1995; Woods et al., 1972).

3. **Brittleness** — Many of the systems introduced in this subchapter did not handle synonyms, paraphrasing, or spelling errors well. Manual adaption / handling was needed in order to becomes resilient against each class of problems (Kate & Mooney, 2006; Yaghmazadeh et al., 2017).

4. **Poor scalability** — With potentially more complex underlying databases, traditional solutions often showed to perform worse. Reddy, Lapata, and Steedman found, that with increasing schema complexity more compute was required to resolve the natural language query to a suitable query candidate making them less transferable and scalable than initially

anticipated (Reddy et al., 2014) — "Evaluating on all domains in Freebase would generate a very large number of queries for which denotations would have to be computed ... Our system loads Freebase using Virtuoso and queries it with SPARQL. Virtuoso is slow in dealing with millions of queries indexed on the entire Freebase, and is the only reason we did not work with the complete Freebase." which indicates underlying system design issues with runtime complexity.

These flaws of traditional NL2SQL approaches made it apparent, that a different class of approaches is needed, which increase transferability and reduce the brittleness since users are "unwilling to trade reliable and predictable user interfaces for intelligent but unreliable ones" according to Popescu et al. (2003). Whilst many approaches outlined tractable ways to increase user satisfaction and accuracy (like Codd did in 1974 with a conversational approach), NLIDBs were and are not considered to be a solved problem.

2.3 Neural NL2SQL Approaches

The previously outlined limitations of traditional approaches to solving NL2SQL / implementing NLIDBs pushed the research branch around neural network application forward to step in and propose new solutions which address the brittleness, transferability and scalability concerns addressed with logical programming approaches. Neural approaches showed to yield significant improvements in terms of transferability and overall accuracy which led to a paradigm shift in this research field.

2.3.1 Early Neural Approaches

In 2017 Zhong, Xiong, and Socher released Seq2SQL which represents a significant breakthrough and leap in NLIDB research. Seq2SQL was an early research system that in the field of neural network application and as one of the first papers to frame the implementation of NLIDBs / NL2SQL Systems as a reinforement learning problem. The system utilized iterative query execution in the reward function to improve its accuracy (Zhong et al., 2017). In the same paper Zhong, Xiong, and Socher introduced WikiSQL, a training dataset, which enables large scale (in 2017) model training.

SQLNet (Xu, Liu, & Song, 2017) addressed primarily the order-sensitivitiy trait of Seq2SQL (Zhong et al., 2017) that was prevalent due to being a derivative approach from sequence-to-sequence approaches. SQLNet diverges from sequence-to-sequence and joins multiple research threads, employing a sketch-based query generation. SQLNet breaks down complex queries into smaller (hence more manageable) sub-queries which can then be individually sketched and refined, yielding a system that outperformed state-of-the-art by 9% to 13% (Xu et al., 2017).

Yu, Li, Zhang, Zhang, and Radev have introduced TypeSQL, a variation of the SQLNet-approach, in 2018. TypeSQL's primary difference to SQLNet is the encoding of type information for SQL generation. The approach scanned for entity references and values in natural language and was able to improve performance by 5.5% over SOTA-Models like SQLNet whilst requiring significantly less training time, indicating that type information was a useful information for deriving accurate SQL queries from user input (Yu, Li, et al., 2018).

2.3.2 Intermediate Neural Developments

Later in 2018 Yu, Yasunaga, et al. released SyntaxSQLNet, a followup research to TypeSQL, which represented a slight change in approach and research focus. In direct comparison SyntaxSQLNet focused primarily around complex query generation using a syntax tree decoder, allowing for longer and more cohesive query generation (Yu, Yasunaga, et al., 2018). This advancement over TypeSQL allowed more complex queries to be reliably generated, enabling multiple clauses aswell as nested queries. SyntaxSQLNet was one of the earlier research efforts which utilized Spider instead of WikiSQL (introduced by Zhong et al. (2017)), a large-scale NL2SQL dataset, incorperating 10.181 hand annotated natural language question and alongside 5.693 unique SQL examples that spread across 138 different domains (Yu, Zhang, et al., 2018). This research led the transition of comparatively simple, research-grade, neural systems for NLIDBs towards systems which are feasible in the real world.

Building on the above approaches, Guo et al. have introduced IRNet, a neural network approach using intermediate representation as a bridge between natural language and SQL in which semantic queries could be expressed. The intermediate format SemQL (or semantic query language) was utilized to transform and synthesize queries on the actual database schema more accurately than Seq2SQL. IRNet followed a three phase approach: schema linking between the natural language query and database layout, synthesis of SemQL as intermediate representation and deterministic conversion of SemQL to SQL. This approach allowed IRNet to outperform state-of-the-art approaches on the Spider benchmark by 19.5%, placing IRNet at an overall accuracy of 46.7% (Guo et al., 2019).

Following IRNet, graph neural networks (GNN) have been explored as alternative architecture by Bogin, Berant, and Gardner (2019), representing the database schema as a graph and using message passing to model relationships between tables, columns and natural language input. This approach demonstrated the capability to improve reasoning and query generation capability. Bogin et al. showed that when evaluating against the Spider benchmark GNN outperforms both SyntaxSQLNet (and therefore SQLNet & TypeSQL). Although presenting a signficiant advancement over previous state-of-the-art approaches, GNN falls behind in performance against IRNet by 6% (Bogin et al., 2019; Guo et al., 2019).

2.3.3 Relation-Aware Transformer Approaches

The release of RAT-SQL (Relation-Aware Transformer for SQL) Wang, Shin, Liu, Polozov, and Richardson (2020) represents the most significant leap in research of neural NL2SQL approaches. RAT-SQL diverged from earlier research through emphasizing the relationship between natural language and the database schema elements using relation-aware self-attention representing a novel approach for solving *schema linking* (Wang et al., 2020).

RAT-SQL's primary innovations was the ability to infer, understand and utilize the relationship between individual tokens in the natural language query and link it to the database schema. Thus allowing for reasoning capabilities on the actual database schema while generating the query.

This architecture yielded a 57.2% in exact match accuracy when being evaluated on the Spider benchmark, substantially outperforming comparative approaches like GNN, IRNet and IRNet V2 by 10.5%, 9.8% and 8.7% respectively. Although overall accuracy improved across all approaches when being paired with BERT (Bidirectional Encoder Representations from Transformers, a popular pre-trained lanugage model from Google) the δ between the indivudual approaches remained relatively steady, leaving RAT-SQL outperforming state-of-the-art approaches by 5% to 12.2% further demonstrating the capability advancement yielded by this system (Wang et al., 2020).

2.3.4 Comparative Analysis of Neural Approaches

The evolution from early neural approaches to RAT-SQL emphasized the rapid advancements that happened in the research field of neural NL2SQL approaches in different dimensions:

1. **Model Complexity** — Given the research progression from early sequence to sequence translation approaches (Seq2SQL) towards sketch based and type augumented and graph based approaches (TypeSQL, SQLNet, GNN) and syntax tree decoding emphasized by SyntaxSQLNet, neural approaches continously advanced in the complexity of approaches that is required to beat state-of-the-art approaches in contemprorary benchmarks like Spider. RAT-SQL presents one of the late and most complex advancements in the field of neural NL2SQL approaches with its adapted self attention mechanism (Bogin et al., 2019; Wang et al., 2020; Yu, Li, et al., 2018; Yu, Yasunaga, et al., 2018; Zhong et al., 2017).

2. **Transferability** — Each of the approaches introduced above represents a succession in terms of their transferability. The field of neural NL2SQL approaches significantly improved the ability for NLIDBs to generalize over the underlying database schemas. RAT-SQL showed the strongest cross-domain accuracy (that is benchmarked by the Spider benchmark). With standard benchmarks emerging it became easier to verify and quantify which approach had the highest transferability as Spider specifically had independent development and test datasets, preventing approaches from over-optimizing on training data (Wang et al., 2020; Yu, Zhang, et al., 2018).

3. **Robustness** — As research systems advanced in complexity and shifted from raw input to output translation (Seq2SQL) their robustness steadily increased. Through more approaches like SyntaxSQLNet which utilized structured decoding, IRNet which relied on an intermediate representation and RAT-SQL the challenges around *schema linking* outlined by Wang et al. (2020) have increasilingly led to more robust systems that can handle rephrasings, spelling mistakes and variations in natural language usage far beyond what traditional NL2SQL approaches could accomplish Guo et al. (2019); Wang et al. (2020); Yu, Yasunaga, et al. (2018).

4. **Query Complexity** — The performance on complex queries involving multiple tables, relying on complex aggregations, nested structures and joins dramatically improved over the course of the research that happened in this field. Whilst IRNet reresents one of the first singificant advancements when it comes to the ability of neural approaches to handle complex queries, RAT-SQL still showed to outperform the intermediate representation approach introduced by IRNet by up to 10.5% (Guo et al., 2019; Wang et al., 2020).

5. **Schema Understanding** — Whilst early approaches like Seq2SQL primarily applied reinforcement learning for end to end query generation (Zhong et al., 2017), later approaches like TypeSQL, GNN and specifically RAT-SQL showed novel and state-of-the-art *schema understanding / schema linking* capabilities, yielding the ability to accurately reason about user intent and traverse the database schema while generating queries (Bogin et al., 2019; Wang et al., 2020; Yu, Li, et al., 2018).

2.3.5 Limitations of Neural Approaches

Despite the dramatic *accuracy, transferability* and *robustness* improvements that could be observed with late neural approaches (Guo et al., 2019; Wang et al., 2020), neural approaches still suffered from serious shortcommings / unsolved challenges:

1. **Training Data** — Utilizing neural networks these approaches required susbtantially more training data (ie. natural language paired with output SQL queries) than traditional systems which required serious efforts of data collection (Yu, Zhang, et al., 2018).

2. **Correctness** — The inherent mismatch between neural networks and formal languages yielded cases where models produced invalid SQL code. Approaches like SyntaxSQLNet improved the tried to solve this circumstance by utilizing syntax trees during decoding but nonetheless syntactic correctness remained a challenge across future iterations of neural systems. (Yu, Yasunaga, et al., 2018)

3. **Domain Language** — Despite increased *transferability* characteristics neural approaches still suffered from a limited vocabulary and inter-domain understanding of terminology and relation between concepts which made highly domain specific natural language queries challenging.

4. **Observability** — The black-box nature of neural networks made approaches relying on them, particularly the ones with complex architectures, hard do understand / explain in case when neural systems yielded undesirable output.

The introduction and advancement of early neural NL2SQL approaches led to significant advancements in the research and feasibility of NLIDBs. The research shift started in this era established the foundations for further and more advanced machine learning approaches (specifically language model oriented approaches)being researched. Neural approaches showed to significant improvements in performance when being paired with pre-trained language models (Wang et al., 2020) which led to further research on their applicability.

2.4 Pre-trained Language Models

The advantages of combining specialized neural networks with general-purpose pre-trained language models led to a pivotal point in the NL2SQL research field towards focusing increasingly on the application of pre-trained language models for NLIDBs. Models like BERT or T5 offer noticable performance improvements (especially when it comes to language understanding) over specialized NL2SQL networks due to training happening on unrestrained amounts of natural language data, instead of pure NL2SQL datasets which are often fairly limited in size and therfore natural language use — Spider2.0 which is a contemprary NL2SQL benchmark consists of just 632 real-world questions (Lei et al., 2025). Thus PLM-based (or at least augumented) NL2SQL systems can observe dramatic performance improvements through the language models's ability to understand patterns and identify semantic relationships of natural language query elements.

2.4.1 Early Pre-trained Language Model Adaptations

The above outlined benefits have led to concrete research efforts focusing on the question whether the sole application of pre-trained language models could outperform neural state-of-the-art approaches — which often implicitly require a far more sophisticate architecture when it comes to natural language analysis.

In the time of emerging PLM applicationm Grappa was introduced by Yu et al. in 2020 — a novel grammar-augmented pre-training approach built on RoBERTa$_{\text{LARGE}}$ (a derivative model from BERT). It generates synthetic training data (ie. natural language and sql pairs) using a synchronous context-free grammar (SCFG) which analyses and identifies patterns in natural language queries that can be used as templates for sythesizing training data. The specialized pre-training helps Grappa to establish a robust connection between natural-language and database schema elements, showing significant improvements on existing approaches on multiple contemporary benchmarks like Spider and WikiSQL (Yu et al., 2020).

Several NL2SQL approaches in this era focused on *schema understanding* and *schema linking* — the generalizability of PLMs required advanced techniques on ensuring that models both understand the semantic intent of users when querying and correctly identify database schema elements in natural language queries. Thus improving semantic accuracy of generated SQL queries. StruG (Structure-Grounded-Pretraining) was introduced in 2020 by Deng et al. and presented a novel pretraining approach that improves model abilities when it comes to *schema linking*, it separates the problem in three facets: column grounding, value grounding and column-value mapping. In direct comparison with Grappa, StruG achieves similar performance while being significantly cheaper to train (Deng et al., 2020).

In parallel, Zhong, Lewis, Wang, and Zettlemoyer released GAZP (Grounded Adaptation for Zero-shot Executable Semantic Parsing) in 2020. Zhong, Lewis, Wang, and Zettlemoyer specifically addressed the challenge of adapting semantic parsers across databases / domains which was a apparent problem with neural approaches which had a strong tendency to overfit on benchmark datasets. Its novel contribution was the combination of forward semantic parsing with a backward utterance generator which allowed for data synthesis in unseen environments which could then be used to adapt the semantic parser (Zhong et al., 2020). This approach enables a improvement in robustness and accuracy in situations where training and inference environments differ without requiring manually annotated examples (Zhong et al., 2020).

2.4.2 Advanced Pre-trained Language Model Approaches

Building on earlier foundational research on PLM application for NL2SQL tasks, researchers have developed increasingly complex systems that leveraged pre-trained language models whilst addressing their limitations when it comes to generating valid SQL.

Choi, Shin, Kim, and Shin introduced Ryansql (Recursively Yielding Annotation Network for SQL) in 2020, which implements a sketch-based approach for decomposing complex SQL generation into multiple smaller problems. Ryansql transformed nested statements into a set of top-level statements using the Statement Position Code (SPC) technique. This flattening of structure allowed Ryansql to limit the complexity of the query generation problem whilst maintaining its ability to answer complex questions by recomposing complex queries from their parts. This approach allowed Ryansql to achieve 58.2% accuracy on the Spider benchmark, representing a 3.2% improvement over contemporary state-of-the-art approaches at the time (Choi et al., 2020). The sketch-based approach makes Ryansql a PLM-augumented successor of SQLNet which was a early neural approach to employ sketch-based query generation (Choi et al., 2020; Xu et al., 2017).

A significant advancement in terms of execution accuracy was reached with the application of T5-Models for NL2SQL tasks. T5 (Text-to-Text Transfer Transformer) Models have proven themselves as well-suited for for query generation — T5-3B for NL2SQL yielded 71.4% execution accuracy and thus presented a breakthrough in this domain of research (Rajkumar et al., 2022). This established a new baseline for PLM-based approaches and demonstrated that general-purpose language models could not only compete but outperform specialized architectures by far when properly fine-tuned (Rajkumar et al., 2022).

Following the adavancements through T5, J. Li, Hui, Cheng, et al. introduced GRAPHIX-T5 in 2023, which combined the T5 PLMs with a further graph-aware layers for NL2SQL tasks. This architecture could leverage both pre-trained knowledge of T5 models aswell as the database schema structure during inference. GRAPHIX-T5 constructs a schema graph where nodes represent tables and columns and edges represent relationships between them, such as foreign keys or columns association. This architecture allows the model to deeply understand relationships and the layout of the database schema (J. Li, Hui, Cheng, et al., 2023). GRAPHIX-T5 outperformed standard T5 models significantly, with GRAPHIX-T5$_{\text{LARGE}}$ showing 6.6% increase in execution accuracy over T5$_{\text{LARGE}}$. When both GRAPHIX and the baseline T5 models were combined with

PICARD (a novel constrained decoding mechanism) absolute δ between them jumped to 7.6% (81.0% in absolute numbers), evaluated on SPIDER-DEV (J. Li, Hui, Cheng, et al., 2023).

In parallel, H. Li, Zhang, Li, and Chen proposed RESDSQL in 2023, which proposed to decouple *schema linking* and *skeleton parsing*. This addressed the typical challenges sequence-to-sequence models faced when simultaneously trying to link both schema elements and generate the query skeleton (e.g. `SELECT <columns> FROM <table>`). RESDSQL further employed a ranking approach to filter schema elements before passing them to the model for query generation, which reduced noise (when working with large database schemas) and enabled passing only the most relevant parts. This twofold approach allowed RESDSQL to achieve state-of-the-art performance when being evaluated on SPIDER, outperforming GRAPHIX-$T5_{3B}$-PICARD by 0.8% in execution accuracy. When combined with NATSQL (a contemporary intermediate representation approach introduced by Gan et al. (2021)) absolute improvement over GRAPHIX-$T5_{3B}$-PICARD jumped to 3.1% emphasizing the robustness gain decoupeled architectures have over model-oriented approaches.

These advancement showed rapid improvements over earlier methods — far surpassing neural approaches — through advanced mechanisms when it comes to schema understanding and query generation. The wide language understanding inherited from PLM-basemodels further strengthens robustness and shows effectiveness through a large gain on the SPIDER benchmark. Collectively these approaches represent a leap in NL2SQL research, emphasizing their usability potential and real-world feasibility. This era primed the research field for the transition towards large language model adoption.

2.4.3 Constrained Decoding and Ranking Techniques

A major challenge in NL2SQL research is making sure that model generated queries are not just semantically accurate but also syntactically valid queries and thus executable. To address this issue Scholak, Schucher, and Bahdanau released PICARD (Parsing Incrementally for Constrained Auto-Regressive Decoding) in 2021, a constrained decoding mechanism for language models which utilizes the SQL grammar and constrained decoding mechanisms to incrementally parse the generate SQL, rejecting invalid tokens based on the grammar. PICARD showed to significantly improve the performance of pre-trained language models (like T5 or BERT) when it comes to NL2SQL tasks, lifting them from mid-level to state-of-the-art solutions on the SPIDER benchmark (Scholak et al., 2021).

PICARD operates as a incremental parser during model output decoding of pre-trained language models and continuously evaluates the probability of each token. Instead of just passing model outputs to a database for execution PICARD incrementally parses and validates the generated SQL, rejecting tokens if neeeded thus significantly improving the valid output accuracy (sometimes referred to as VA) of language models. This approach is addressing a significant issue associated with pre-trained language models — while they outperform in natural language understanding and reasoning, they often lack SQL grammer knowledge and tend to generate queries that are not executable due to ther unconstrained output space (Scholak et al., 2021).

The above introduces RESDSQL built ontop of PICARD's foundations and used a ranking-enhanced framework for input encoding. These two approaches represent a unqiue class of approaches that utilize input and output constraining in order to increase the performance characteristics of pre-trained language models (H. Li et al., 2023).

2.4.4 Advantages of PLM Approaches

PLM approaches to NL2SQL tasks have yielded significant performance improvements for the NL2SQL domain and represent a leap in NLIDB-research. They primed the research field towards using language models which led to a transition towards large language models in the following years. Namely PLM approaches brought a series of upsides with them:

1. **Compute Efficiency** — PLMs like RESDSQL achieve high accuracy (up to 84.1% on Spider depending on variants) wilst using far fewer parameters than conteporary LLMs, making them significantly more efficient and therefore reduce hardware requirements for their deployment (H. Li et al., 2023).

2. **Transferability** — Approaches like Grappa and StruG can incorporate domain-specific understanding of natural language, table structures and SQL syntax during pre-training which addressed one of the primary issues with neural approaches (Deng et al., 2020; Yu et al., 2020).

3. **Vocabulary** — PLMs offer a larger vocabulary due to the vast amounts of training data available. This enables them to handle a wide variety of natural language patterns which addresses the benchmark-overfitting tendency of neural approaches which primarily trained on the development sets of contemporary benchmarks.

2.4.5 Limitations of PLM Approaches

Although representing the state-of-the-art at the time, PLMs introduce a class of problems which are associated with their non-NL2SQL associated nature. There have been an array of approaches to mitigate these shortcommings but nonetheless they must be considered when using a PLM-based approach to NL2SQL:

1. **Fine-tuning Requirements** — Most PLMs require substantial domain-specific, or at least NL2SQL specific, fine-tuning, limiting a straight forward adaptation to new domains or databases. Although being significantly more efficient than LLM-based approaches the potential need for initial fine-tuning represent a significant computational resource burden. Furthermore when not using synthetic data generation (e.g. Grappa) annotated datasets of training data are needed to achieve appropiate performance characteristics (Yu et al., 2020).

2. **Wide Input & Output Space** — Due to the general nature of PLMs their input and output space is often far larger than needed NL2SQL tasks. "Large pre-trained language models for textual data have an unconstrained output space; at each decoding step, they can produce any of 10,000s of sub-word tokens" (Scholak et al., 2021). This applies to both the input and output token space, therefore multiple approaches have been researched which focus on constraining these to the subset needed for NL2SQL tasks. Namely GRAPHIX-T5 and Picard have proposed potential (and promising) solutions to this issue (J. Li, Hui, Cheng, et al., 2023; Scholak et al., 2021).

3. **Limited Schema Awareness** — Due to being general purpose, and non-NL2SQL optimized, PLMs tend to incorporate limited amounts of schema awareness when being applied out of the box for NL2SQL tasks. Multiple research efforts focused on improving this situation, most notably Resdsql and GRAPHIX-T5 tried to improve the schema linking & awareness of PLMs (H. Li et al., 2023; J. Li, Hui, Cheng, et al., 2023), nonetheless the non-specialized nature of PLMs prevents NL2SQL being part of the fundamental model architecture.

These characteristics positioned PLMs as powerful but comparatively resource-intensive solutions for NL2SQL (especially in direct comparison with neural approaches), ultimately yielding the research domain to transition toward exploring Large Language Model approaches that promise even greater flexibility in adaptation and potentially superior handling of complex queries through advanced in-context learning approaches.

2.4.6 Comparison with Large Language Models

The research on applying pre-trained language models for NL2SQL tasks primed the field for the transition towards LLM usage. While PLMs like T5 and BERT range from millions to a few billion parameters, prevalent LLMs such as GPT-3 and GPT-4 operate at significantly larger scales, ranging from a few billions to hundred of billions parameters. The scale of LLMs enables in-context learning techniques that enable significantly easier and cheaper transferability of NL2SQL systems across domains (D. Gao et al., 2023). The δ of deployment, inference and training requirements of these two approaches are significant due to the size difference in models, which transfer to hardware requirements and therefore cost. While PLMs can require extensive fine-tuning on domain-specific data which may aswell be resource intensive (Deng et al., 2020; H. Li et al., 2023; J. Li, Hui, Cheng, et al., 2023; Yu et al., 2020), LLMs transfer the cost to the inference environment, where model modificants are less impactful, due to the extensive pre-training that took place. Approaches like DinSQL show that with the application of LLMs the engineering challenges around model instruction gained relevance while model training became less of a central problem to solve (Pourreza & Rafiei, 2023).

2.5 Large Language Models

The emergence of LLMs such as GPT-3, GPT-4, and Claude fundamentally transformed the landscape of NL2SQL research. Early experiments with LLMs for NL2SQL tasks showed state-of-the-art capabilities in comparsion with contemporary PLM approaches (D. Gao et al., 2023). Rajkumar et al. (2022) demonstrated that Codex (a contemporary model based on GPT-3), without any fine-tuning efforts, could achieve competitive performance on Spider, outperforming many state-of-the-art approaches that required extensive training. This breakthrough challenged the contemporary assumption that further specialization of model architectures would yield increases in NL2SQL performance (e.g. Graphix-T5) (J. Li, Hui, Cheng, et al., 2023).

2.5.1 In-Context Learning

In-Context Learning (ICL) is a foundational approach for leveraging the ability of LLMs to utilize larger context windows for inference than traditional PLMs. Typical context windows of state-of-the-art LLMs can reach up to hundred thousands of tokens. This characteristic of LLMs enabled researchers to utilize this context window to provide examples of accurate NL2SQL translation instead of applying parameter updates. This paradigm shift has made developing NL2SQL systems significantly more accessible.

The fundamental principle of few-shot learning for NL2SQL involves providing the LLM with a small number of example pairs of natural language and their corresponding SQL representation. These examples can benefit the model's understanding of mapping between natural language and SQL syntax. This essentially builds on top of prior research like Grappa and StruG, but applying these examples at inference time, rather than training time. Although this increases the inference cost of such a system, the upsides lie primarily in the flexibility of such an approach — database content / prior usage of the system can be dynamically utilized, rather than requiring retraining.

Example selection strategies showed to have a considerable impact on ICL performance. D. Gao et al. (2023) evaluated various example selection methods like *Random*, *Question Similarity Selection (QTS)*, *Masked Question Similarity Selection (MQS)*, and *Query Similarity*

Selection (QRS). D. Gao et al. propose a novel strategy to select, organize and present ICL examples to LLMs. DAIL-SQL utilizes both question and query similarity, masking domain-specific words and prioritizing examples that exceed a similarity threshold of τ (D. Gao et al., 2023, p. 5). DAIL-SQL encodes examples as question-SQL-pairs without the respective schema to improve token efficiency. Using a Code Representation Prompt (CR) for question and schema encoding yielded DAIL-SQL to achieve state-of-the-art 86.6% execution accuracy on SPIDER.

The comparison between zero-shot and few-shot performance reveals the accuracy gain potential through supplying examples to models in the inference context. While contemporary LLMs (such as GPT-4) have demonstrate impressive zero-shot performance (achieving 72.3% execution accuarcy on benchmarks like SPIDER) (D. Gao et al., 2023, Table 1, p. 8), few-shot learning still shows to substantially improve model performance. D. Gao et al. (2023) shows that even one-shot learning boosts GPT-4's execution accuracy to 80.2%, representing a 7.9% increase, while five-shot learning reaches 82.4% (D. Gao et al., 2023, Table 2, p. 8).

Especially with complex queryies which can involve multiple tables, nested queries and complex joins, zero-shot approaches often dramatically underperform k-shot ones. NL2SQL approaches that dont supply examples to the model during inference time fail more frequently to generate semantically accurate SQL queries (D. Gao et al., 2023). Notable is the leap in exact match ratio measured by the SPIDER benchmark — jumping from 22.1% for GPT-4 using zero-shot to 71.9% with five-shot. The results presented by D. Gao et al. (2023) show signficant correlation between k and the execution accuracy of k-shot approaches.

This effectiveness has established ICL approaches as a standard technique applied in LLM-based NL2SQL approaches. Contemporary approaches like XIYAN-SQL, CHASE-SQL AND DIN-SQL all utilize variations of ICL to achieve state-of-the-art results (Y. Gao et al., 2025; Pourreza et al., 2024; Pourreza & Rafiei, 2023).

2.5.2 Self-Correction and Iterative Refinement

Pourreza and Rafiei (2023) proposed DIN-SQL as an innovative approach to NLIDBs that rely on LLMs. DIN-SQL decomposes complex queries into sub-parts and utilizes in-context-learning and self-correction during the generation phase. Compared to DAIL-SQL which relies on example selection during the in-context-learning phase, DIN-SQL focuses on a refinement loop that allows the model to self- correct errors it made during the initial generation phase — thus the model can repair schema linking, syntactic or semantic errors. By explicitly instructing the LLM to review its work against a specfic schema, the user input and potential database errors, DIN-SQL achieves a high execution accuracy on SPIDER with 85.3%. Therefore DIN-SQL outperforms contemporary approaches but is surpassed by by DAIL-SQL by 1.3% (D. Gao et al., 2023; Pourreza & Rafiei, 2023). Furthermore DIN-SQL makes observations on the impact that the self-correction prompt can steer results significantly — Pourreza and Rafiei found that using *generic self-correction* (ie. assuming the query contains errors) lowers the execution accuracy by 4.2% on SPIDER compared to *gentle self-correction* (ie. assuming nothing about the validity of the query). It was noted that the impact of the self-correction mechanism relies on the model size, with smaller models performing better with *generic self-correction* and larger models performing better with *gentle self-correction* (Pourreza & Rafiei, 2023). The self-correcting nature of DIN-SQL represents a diversion from DAIL-SQL's emphasis on input optimization towards output refinement. Pourreza and Rafiei demonstrate how structured introspection can play a significant role in enhancing LLM performance for formal language generation tasks.

Building upon DIN-SQL's self-correction module, Askari, Poelitz, and Tang (2024) proposed MAGIC (Multi-Agent Guideline for In-Context Text-to-SQL), which further advances the self-correction mechanism through harnessing a set of specialized agents to automate the self-correction prompt engineering (Askari et al., 2024). MAGIC consists of a manager agent, a correction agent and a feedback agent that collaboratively refine LLM instructions during the refinement loop. Further MAGIC derives common faliure patterns of the initial query generation phase from training data, allowing it to efficiently spot the most common mistakes that the model makes at generation time. This approach represent a further advancement on Pourreza and Rafiei's DIN-SQL, effectively supersetting the *generic* and *gentle* correction mechanisms through an intelligent, self-adapting one (Askari et al., 2024). The autogenerated guidelines from MAGIC yield 85.6% execution accuracy on the Spider development set — representing a 5.31% improvement over DIN-SQL's human written correction guidelines. These results emphasize that optimized self-correction mechanisms have the ability to significantly drive up overall system performance of NLIDBs (Askari et al., 2024).

While DIN-SQL and MAGIC focus on automated self-correction in single-turn settings, H. Zhang, Cao, Xu, Chen, and Yu (2024) introduced the concept of Chain-of-Editions (CoE-SQL), which addresses the unique challenges of multi-turn conversational NLIDBs. Conversational interfaces for NL2SQL systems enable human-in-the-loop refinement. Interactive information seeking from the user has shown to be an effective way to drive overall accuracy of the system and improve user satisfaction (F. Li & Jagadish, 2014). Rather than approaching each query independently, CoE-SQL recognizes that in a conversational context, successive SQL queries usuallly require only small and incremental modifications of the previous queries. Interactive user input is an effective measure for dealing with ambiguous natural language queries (F. Li & Jagadish, 2014; Montgomery, 1972; H. Zhang et al., 2024).

2.5.3 Candidate Selection Frameworks

Contemporary NL2SQL approaches have increasingly emphasized on the generation of query candidates and their selectiion as a promising architecture. Candidate selection strategies have shown significant performance improvements on challenging benchmarks. These approach acknowledge the inherent difficulty of generating perfect SQL queries in one attempt / using one generation mechanism, even with capable LLMs and modern self-correction mechanisms.

Pourreza et al. (2024) introduced CHASE-SQL in 2024, a framework that leverages multiple reasoning paths to generate multiple query candidates. After the initial generation phase CHASE selects the most promising solution to the natural query input. CHASE-SQL harnesses three different generation strategies: A divide-and-conquer approach which breaks down complex natural language queries into multiple sub tasks that can be individually tackled, a chain-of-thought based generation approach which inspects execution plans of SQL queries and a schema-aware generation of synthetic examples that can be used for in-context learning (Pourreza et al., 2024). These different generation mechanisms produce a set of query candidates that each have different characteristics. For candidate selection CHASE harnesses a fine-tuned LLM that can do binary selection of candidates. Pourreza et al. have demostrated that their query selection approach is more robust than apparent alternatives and yields state-of-the-art performance with 73% execution accuracy on Bird and 87.6% on Spider (Pourreza et al., 2024).

Conceptually similar work has been done by Y. Gao et al. with XiYan-SQL which is architected as multi-generation ensemble strategy with better schema representation. XiYan-SQL integrated in-context learning alongside supervised fine-tuning approaches to generate query candidates (Y. Gao et al., 2025). A key contribution of Y. Gao et al. (2025) is their M-Schema representation of database schemas, which improves the models schema awareness and reduces frequent schema linking errors. XiYan-SQL enhances accuracy by utilizing multiple different strategies that have complementary characteristics during query generation to enhance the robustness of the overall system. The query generation stage utilizes both a fine-tuned SQL generation model aswell as ICL strategies to achieve a breadth of candidate coverage. Following to the query generation stage a self-correction stage (referred to as *refinement* stage by Y. Gao et al.) is utilized to correct common errors. Lastly a selection model is used to choose the most accurate candidate that was produced during the generation stage (Y. Gao et al., 2025). Through this sophisticated and diverse architecture XiYan-SQL was able to achieve impressive results across contemporary NL2SQL benchmarks — achieving 89.65% execution accuracy on Spider and 73.34% on Bird, which renders XiYan-SQL state-of-the-art (Y. Gao et al., 2025).

Both CHASE and XiYan-SQL show that diversifing candidate generation and training specialized models for candidate selection yield state-of-the-art execution accuracy which significantly outperforms single-path generation approaches. The success of these two approaches indicates that for increasingly complex NL2SQL tasks (such as Spider2.0), the capability to generate multiple valid interpretations of the natural language query is an important stepping stone to achieving meaningful execution accuracy. Both CHASE and XiYan-SQL rely on specialized candidate selection models which renders these approaches to combine the strengths of LLMs when it comes to language understanding and transferability with the robustness of specialized model architectures for candidate ranking and selection.

2.5.4 Retrieval-Augumented Generation

Retrieval-Augmented Generation (RAG) has emerged as a powerful paradigm for enhancing NL2SQL systems by integrating external knowledge retrieval with the generative capabilities of LLMs. This technique is seen in all above introduced papers in varying forms (Askari et al., 2024; D. Gao et al., 2023; Y. Gao et al., 2025; Pourreza et al., 2024; Pourreza & Rafiei, 2023; H. Zhang et al., 2024). The most prevalent form of RAG in NL2SQL is the encoding of the database schema into the LLM prompt for in-context-learning. This allows LLMs to be aware of the table structures and names, foreign key relationships, primary keys etc. Y. Gao et al. (2025) proposed the serialization of database schemas in the M-Schema format, a semi structured, text-based schema description. D. Gao et al. (2023) proposed to encode the schema using a Code Representation Prompt (CR) which refers to the encoding of the raw SQL statements need to construct the schema. Y. Gao et al. (2025) provided an ablation study for the M-Schema which yielded questionable results on the optimization of this approach. While the M-Schema format yielded best results when XiYan-SQL was combined with GPT-4o or Claude 3.5 Sonnet, it performed worse than alternatives on DeepSeek and Gemini models (Y. Gao et al., 2025).

A relevant optimization technique for RAG is the selection of a schema subset before encoding the schema for the model. Resdsql was one of the earlier approaches to explore subset-encoding of database schemas with Shen et al. (2024) building on top of this (H. Li et al., 2023; Shen et al., 2024). Shen et al. (2024) introduced ASTReS which dynamically retrieves database schemas and uses abstract syntax trees (ASTs) to select optimal few-shot examples for ICL. By pruning the ASTs down to the most relevant subset, ASTReS achieved the highest at-the-time (2024) execution accuracy on Spider with 86.6% indicating that subset-encoding is a sensible optimization mechanism. ASTReS was combined with GRAPHIX-T5 in order to achieve this result (Shen et al., 2024).

The impact of RAG when NL2SQL systems face large and complex database schemas has been particularly significant. Traditional approaches struggle when database schemas contain hundreds or thousands of tables and columns, as the complete schema may not fit within model context windows depending on their size. RAG-based systems address this by dynamically retrieving only the most relevant portions of the schema based on the natural language query. The technique of subset-encoding becomes especially relevant in enterprise environments where database schemas can be extremely large and complex. Recent benchmarks like Spider2.0 emphasize enterprise environments and show that existing solutions underperform in those scenarios, often reaching single digit execution accuracy.

The work of Shen et al. indicates that prefiltering of the environment of language models is an effective and promising technique that has the ability to reduce computational requirements of contemporary NL2SQL systems. As introduced above, recent state-of-the-art systems often utilize closed-source models like Gemini, GPT-4(o), Claude 3.5/3.7 Sonnet etc which often come with massive parameter sizes (reaching hundreds of billions of parameters). ASTReS demonstrated that efficient schema retrieval mechanisms enable smaller models (e.g. GRAPHIX-T5) to achieve state-of-the-art performance against LLM based approaches like DAIL-SQL (D. Gao et al., 2023; Shen et al., 2024).

2.6 Research Gaps

2.6.1 Deployment Gaps

3 Theoretical Foundations

3.1 Problem Decomposition

3.2 Requirements

3.3 System Design

3.3.1 Architecture Design

3.4 Technical Implementation Strategies

4 Implementation

4.1 Development Environment and Tools

4.2 Integration of the Model

4.3 Development of the PostgreSQL Extension

4.4 Optimization

5 Evaluation

5.1 Test Environment and Methodology

5.2 Performance Tests

5.2.1 Latency

5.2.2 Throughput

5.2.3 Scalability

5.3 Use Cases

5.3.1 Natural Language Queries

5.3.2 Text Generation Within the Database

5.3.3 Semantic Search and Text Classification

5.4 Ablation Study

5.5 Comparison with Alternative Approaches

6 Discussion

6.1 Interpretation of Results

6.2 Limitations of the Implementation

6.3 Ethical and Data Privacy Considerations

6.4 Potential Future Developments

7 Summary and Outlook

7.1 Summary of Results

7.2 Addressing the Research Questions

7.3 Outlook for Future Research and Development

Appendix

Installation Guide

API Documentation

Code Examples

Test Data and Results

References

Androutsopoulos, I., Ritchie, G. D., & Thanisch, P. (1995). Natural language interfaces to databases - an introduction. *CoRR*, *cmp-lg/9503016*. Retrieved from `http://arxiv.org/abs/cmp-lg/9503016`

Askari, A., Poelitz, C., & Tang, X. (2024). *Magic: Generating self-correction guideline for in-context text-to-sql.* Retrieved from `https://arxiv.org/abs/2406.12692`

Bogin, B., Berant, J., & Gardner, M. (2019, July). Representing schema structure with graph neural networks for text-to-SQL parsing. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 4560–4565). Florence, Italy: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/P19-1448/` doi: 10.18653/v1/P19-1448

Choi, D., Shin, M. C., Kim, E., & Shin, D. R. (2020). *Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases.* Retrieved from `https://arxiv.org/abs/2004.03125`

Codd, E. F. (1974, January). Seven steps to rendezvous with the casual user. In J. W. Klimbie & K. L. Koffeman (Eds.), *Ifip working conference data base management* (p. 179-200). North-Holland. Retrieved from `http://dblp.uni-trier.de/db/conf/ds/dbm74.html#Codd74` (IBM Research Report RJ 1333, San Jose, California)

Deng, X., Awadallah, A. H., Meek, C., Polozov, O., Sun, H., & Richardson, M. (2020). Structure-grounded pretraining for text-to-sql. *CoRR*, *abs/2010.12773*. Retrieved from `https://arxiv.org/abs/2010.12773`

Gan, Y., Chen, X., Xie, J., Purver, M., Woodward, J. R., Drake, J. H., & Zhang, Q. (2021). Natural SQL: making SQL easier to infer from natural language specifications. *CoRR*, *abs/2109.05153*. Retrieved from `https://arxiv.org/abs/2109.05153`

Gao, D., Wang, H., Li, Y., Sun, X., Qian, Y., Ding, B., & Zhou, J. (2023). *Text-to-sql empowered by large language models: A benchmark evaluation.* Retrieved from `https://arxiv.org/abs/2308.15363`

Gao, Y., Liu, Y., Li, X., Shi, X., Zhu, Y., Wang, Y., ... Li, Y. (2025). *A preview of xiyan-sql: A multi-generator ensemble framework for text-to-sql.* Retrieved from `https://arxiv.org/abs/2411.08599`

Guo, J., Zhan, Z., Gao, Y., Xiao, Y., Lou, J.-G., Liu, T., & Zhang, D. (2019, July). Towards complex text-to-SQL in cross-domain database with intermediate representation. In A. Korhonen, D. Traum, & L. Màrquez (Eds.), *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 4524–4535). Florence, Italy: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/P19-1444/` doi: 10.18653/v1/P19-1444

Hendrix, G. G., Sacerdoti, E. D., Sagalowicz, D., & Slocum, J. (1978, June). Developing a natural language interface to complex data. *ACM Trans. Database Syst.*, *3*(2), 105–147. Retrieved from `https://doi.org/10.1145/320251.320253` doi: 10.1145/320251.320253

Kate, R. J., & Mooney, R. J. (2006, July). Using string-kernels for learning semantic parsers. In N. Calzolari, C. Cardie, & P. Isabelle (Eds.), *Proceedings of the 21st international conference on computational linguistics and 44th annual meeting of the association for computational linguistics* (pp. 913–920). Sydney, Australia: Association for Computational Linguistics. Retrieved from `https://aclanthology.org/P06-1115/` doi: 10.3115/1220175.1220290

Lei, F., Chen, J., Ye, Y., Cao, R., Shin, D., Su, H., ... Yu, T. (2025). *Spider 2.0: Evaluating language models on real-world enterprise text-to-sql workflows.* Retrieved from `https://arxiv.org/abs/2411.07763`

Li, F., & Jagadish, H. V. (2014). Nalir: an interactive natural language interface for querying relational databases. In *Proceedings of the 2014 acm sigmod international conference on management of data* (p. 709–712). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/2588555.2594519` doi: 10.1145/2588555.2594519

Li, H., Zhang, J., Li, C., & Chen, H. (2023). *Resdsql: Decoupling schema linking and skeleton parsing for text-to-sql.* Retrieved from `https://arxiv.org/abs/2302.05965`

Li, J., Hui, B., Cheng, R., Qin, B., Ma, C., Huo, N., ... Li, Y. (2023). *Graphix-t5: Mixing pre-trained transformers with graph-aware layers for text-to-sql parsing.* Retrieved from `https://arxiv.org/abs/2301.07507`

Li, J., Hui, B., Qu, G., Yang, J., Li, B., Li, B., ... Li, Y. (2023). *Can llm already serve as a database interface? a big bench for large-scale database grounded text-to-sqls.* Retrieved from `https://arxiv.org/abs/2305.03111`

Montgomery, C. A. (1972). Is natural language an unnatural query language? In *Proceedings of the acm annual conference - volume 2* (p. 1075–1078). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/800194.805902` doi: 10.1145/800194.805902

Popescu, A.-M., Etzioni, O., & Kautz, H. (2003). Towards a theory of natural language interfaces to databases. In *Proceedings of the 8th international conference on intelligent user interfaces* (p. 149–157). New York, NY, USA: Association for Computing Machinery. Retrieved from `https://doi.org/10.1145/604045.604070` doi: 10.1145/604045.604070

Pourreza, M., Li, H., Sun, R., Chung, Y., Talaei, S., Kakkar, G. T., ... Arik, S. O. (2024). *Chase-sql: Multi-path reasoning and preference optimized candidate selection in text-to-sql.* Retrieved from `https://arxiv.org/abs/2410.01943`

Pourreza, M., & Rafiei, D. (2023). Din-sql: decomposed in-context learning of text-to-sql with self-correction. In *Proceedings of the 37th international conference on neural information processing systems.* Red Hook, NY, USA: Curran Associates Inc.

Rahaman, A., Zheng, A., Milani, M., Chiang, F., & Pottinger, R. (2024). *Evaluating sql understanding in large language models.* Retrieved from `https://arxiv.org/abs/2410.10680`

Rajkumar, N., Li, R., & Bahdanau, D. (2022). *Evaluating the text-to-sql capabilities of large language models.* Retrieved from `https://arxiv.org/abs/2204.00498`

Reddy, S., Lapata, M., & Steedman, M. (2014). Large-scale semantic parsing without question-answer pairs. *Transactions of the Association for Computational Linguistics*, *2*, 377–392. Retrieved from `https://aclanthology.org/Q14-1030/` doi: 10.1162/tacl_a_00190

Scholak, T., Schucher, N., & Bahdanau, D. (2021). PICARD: parsing incrementally for constrained auto-regressive decoding from language models. *CoRR*, *abs/2109.05093*. Retrieved from `https://arxiv.org/abs/2109.05093`

Shen, Z., Vougiouklis, P., Diao, C., Vyas, K., Ji, Y., & Pan, J. Z. (2024). *Improving retrieval-augmented text-to-sql with ast-based ranking and schema pruning.* Retrieved from `https://`

arxiv.org/abs/2407.03227

Tang, L. R., & Mooney, R. J. (2001). Using multiple clause constructors in inductive logic programming for semantic parsing. In *Proceedings of the 12th european conference on machine learning* (p. 466–477). Berlin, Heidelberg: Springer-Verlag.

Wang, B., Shin, R., Liu, X., Polozov, O., & Richardson, M. (2020, July). RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers. In D. Jurafsky, J. Chai, N. Schluter, & J. Tetreault (Eds.), *Proceedings of the 58th annual meeting of the association for computational linguistics* (pp. 7567–7578). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2020.acl-main.677/ doi: 10.18653/v1/2020.acl-main.677

Woods, W. A., Kaplan, R., & Nash-Webber, B. (1972). *The lunar sciences natural language information system: Final report.* Cambridge, Massachusetts: Bolt, Beranek and Newman, Inc.

Xu, X., Liu, C., & Song, D. (2017). Sqlnet: Generating structured queries from natural language without reinforcement learning. *CoRR*, *abs/1711.04436*. Retrieved from http://arxiv.org/abs/1711.04436

Yaghmazadeh, N., Wang, Y., Dillig, I., & Dillig, T. (2017, October). Sqlizer: query synthesis from natural language. *Proc. ACM Program. Lang.*, *1*(OOPSLA). Retrieved from https://doi.org/10.1145/3133887 doi: 10.1145/3133887

Yu, T., Li, Z., Zhang, Z., Zhang, R., & Radev, D. (2018, June). TypeSQL: Knowledge-based type-aware neural text-to-SQL generation. In M. Walker, H. Ji, & A. Stent (Eds.), *Proceedings of the 2018 conference of the north American chapter of the association for computational linguistics: Human language technologies, volume 2 (short papers)* (pp. 588–594). New Orleans, Louisiana: Association for Computational Linguistics. Retrieved from https://aclanthology.org/N18-2093/ doi: 10.18653/v1/N18-2093

Yu, T., Wu, C., Lin, X. V., Wang, B., Tan, Y. C., Yang, X., … Xiong, C. (2020). Grappa: Grammar-augmented pre-training for table semantic parsing. *CoRR*, *abs/2009.13845*. Retrieved from https://arxiv.org/abs/2009.13845

Yu, T., Yasunaga, M., Yang, K., Zhang, R., Wang, D., Li, Z., & Radev, D. (2018, October-November). SyntaxSQLNet: Syntax tree networks for complex and cross-domain text-to-SQL task. In E. Riloff, D. Chiang, J. Hockenmaier, & J. Tsujii (Eds.), *Proceedings of the 2018 conference on empirical methods in natural language processing* (pp. 1653–1663). Brussels, Belgium: Association for Computational Linguistics. Retrieved from https://aclanthology.org/D18-1193/ doi: 10.18653/v1/D18-1193

Yu, T., Zhang, R., Yang, K., Yasunaga, M., Wang, D., Li, Z., … Radev, D. R. (2018). Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-sql task. *CoRR*, *abs/1809.08887*. Retrieved from http://arxiv.org/abs/1809.08887

Zelle, J. M., & Mooney, R. J. (1996). Learning to parse database queries using inductive logic programming. In *Proceedings of the thirteenth national conference on artificial intelligence - volume 2* (p. 1050–1055). AAAI Press.

Zhang, B., Ye, Y., Du, G., Hu, X., Li, Z., Yang, S., … Mao, H. (2024). *Benchmarking the text-to-sql capability of large language models: A comprehensive evaluation.* Retrieved from https://arxiv.org/abs/2403.02951

Zhang, H., Cao, R., Xu, H., Chen, L., & Yu, K. (2024). *Coe-sql: In-context learning for multi-turn text-to-sql with chain-of-editions.* Retrieved from https://arxiv.org/abs/2405.02712

Zhong, V., Lewis, M., Wang, S. I., & Zettlemoyer, L. (2020, November). Grounded adaptation for zero-shot executable semantic parsing. In B. Webber, T. Cohn, Y. He, & Y. Liu (Eds.), *Proceedings of the 2020 conference on empirical methods in natural language processing (emnlp)* (pp. 6869–6882). Online: Association for Computational Linguistics. Retrieved from https://aclanthology.org/2020.emnlp-main.558/ doi: 10.18653/v1/2020.emnlp-main.558

Zhong, V., Xiong, C., & Socher, R. (2017). Seq2sql: Generating structured queries from natural language using reinforcement learning. *CoRR*, *abs/1709.00103*. Retrieved from http://arxiv.org/abs/1709.00103