Schema-Aware Natural Language to SQL Conversion for Academic Data Management Using Large Language Models with Multi-Tier Fallback

[1]Sujal Yogesh Patil
Information Technology
International Institute of Information Technology
Pune, India
sujalpat1812@gmail.com

[2]Yash Patil
Information Technology
International Institute of Information Technology
Pune, India
yashpatil4648@gmail.com

[3]Aryan Tilekar
Information Technology
International Institute of Information Technology
Pune, India
Atilekar2004@gmail.com

[4]Chaitanya Gite
Information Technology
International Institute of Information Technology
Pune, India
chaitanyagite0024@gmail.com

[5]Prof. Sharda Thete (Guide)
Information Technology
International Institute of Information Technology
Pune, India
shardat@isquareit.edu.in

Abstract : Accessing institutional data remains challenging for non-technical users due to the complexity of Structured Query Language (SQL). This paper presents a schema-aware natural language to SQL (NLP-to-SQL) system designed for academic data management, enabling users to query databases using plain English. The system supports two domains: student records and faculty IQAC data. It leverages Google Gemini 1.5 Pro for SQL generation, enhanced by dynamic PRAGMA-based schema introspection and domain-specific prompt engineering. A four-tier fallback mechanism (Gemini, Claude, GPT-4, and rule-based mock) ensures reliability under API failures. Security is enforced through parameterized queries and identifier validation. Experimental evaluation on 24 queries achieved 100% SQL generation and execution success, with a median latency of 1,838 ms. The system enables rapid adaptation to new domains without retraining, offering a scalable and practical solution for academic institutions.

I.INTRODUCTION

Access to structured data is a fundamental requirement for effective decision-making in academic institutions. However, most institutional databases rely on Structured Query Language (SQL), which requires technical expertise that many educators, administrators, and academic coordinators do not possess. As a result, even simple queries—such as identifying students with low attendance or retrieving faculty development records—often require intervention from technical personnel, leading to delays and reduced operational efficiency.

This problem is particularly significant in academic environments where data is continuously generated and must be analyzed for quality assurance, accreditation processes, and performance monitoring. Although business intelligence tools provide partial solutions, they are often expensive and not specifically designed for academic workflows.

Recent advancements in large language models (LLMs) have enabled natural language interfaces capable of translating user queries into SQL statements. However, many existing systems are limited to benchmark datasets and do not address practical challenges such as dynamic schema adaptation, multi-database environments, and system reliability.

To address these limitations, this paper proposes a schema-aware natural language to SQL system tailored for academic data management. The system dynamically extracts database schema information at runtime, incorporates domain-specific context, and utilizes a multi-tier fallback mechanism to ensure consistent performance. By enabling users to interact with databases using natural language, the system reduces dependency on technical expertise and improves accessibility to institutional data.

NEED OF THE STUDY.

In modern academic institutions, large volumes of structured data—such as student performance records, attendance logs, and faculty activity reports—are continuously generated and stored in relational databases. However, access to this data is largely restricted to users with knowledge of Structured Query Language (SQL). Most educators, administrators, and quality assurance personnel lack this technical expertise, creating a dependency on IT staff for even simple data retrieval tasks. This dependency introduces delays, reduces operational efficiency, and limits timely decision-making.

Existing solutions such as Tableau and Microsoft Power BI offer user-friendly interfaces but are often costly and not tailored to the specific workflows of academic institutions, such as attendance monitoring or IQAC reporting. Moreover, traditional research in natural language to SQL systems primarily focuses on static benchmark datasets and does not address real-world deployment challenges like schema variability, multi-database environments, and system reliability.

Therefore, there is a clear need for a **schema-aware, cost-effective, and reliable natural language interface** that enables non-technical users to interact directly with institutional databases. Such a system can democratize data access, reduce dependency on technical intermediaries, and support faster, data-driven decision-making in academic environments.

3.1Population and Sample

Population

The population for this study comprises all structured academic data maintained within institutional database systems. This includes student academic records (such as marks, attendance, and enrollment details) and faculty activity records (such as FDP participation, MOOC certifications, and research publications). Conceptually, the population represents the complete set of data entities and records that could be queried using a natural language interface in an academic environment.

Sample

The sample used for system design, implementation, and evaluation consists of two real-world institutional datasets:

- **Student Dataset (college_data.db):**
  - 166 students
  - 14,969 records across four tables (students, cce_marks, attendance, attendance_overview)
  - Covers academic performance, continuous evaluation marks, and attendance tracking
- **Faculty Dataset (faculty.db):**
  - 28 faculty members
  - 114 records across six tables (faculty_master, fdp_training, mooc_certifications, journal_publications, etc.)
  - Covers IQAC-related activities such as training programs, certifications, and publications

Additionally, a **benchmark set of 24 natural language queries** was used as a sample for evaluating system performance, covering different query types such as filtering, aggregation, and joins.

3.2 Data and Sources of Data

Data Description

The study utilizes structured academic data representing two primary institutional domains: student academic records and faculty IQAC activity records. The data is organized in relational format and stored in two independent SQLite databases, enabling domain-specific querying and evaluation.

- **Student Academic Data:** This dataset includes detailed records of student performance and attendance. It consists of 166 students with a total of 14,969 records distributed across four tables:
  - *students*: basic student information (roll number, name, division, batch)
  - *cce_marks*: Continuous Comprehensive Evaluation (CCE-I and CCE-II) marks across subjects
  - *attendance*: subject-wise attendance records (lectures conducted vs attended)
  - *attendance_overview*: aggregated attendance percentages
- **Faculty IQAC Data:** This dataset contains faculty activity records relevant to institutional quality assurance processes. It includes 28 faculty members with 114 records across six tables:
  - *faculty_master*: faculty details (name, department, designation)
  - *fdp_training*: faculty development programs (title, institute, duration, level)
  - *mooc_certifications*: online course completions
  - *journal_publications*, *conference_publications*, *book_publications*: research output records

Sources of Data

The data used in this study is derived from official institutional records of an academic institution (IIIT Pune) and processed into structured formats suitable for database integration.

- **Student Data Sources:**
  - Excel spreadsheets (.xlsx) exported from the academic examination system
  - Files include CCE-I and CCE-II marksheets and attendance registers for SE Division A and B
  - Processed using Python libraries such as *openpyxl* and *pandas* before being loaded into SQLite
- **Faculty Data Sources:**
  - CSV files derived from IQAC audit reports (Sections 7b, 7c, 7d, 7f)
  - Includes records of FDP participation, MOOC certifications, and research publications
  - Processed and normalized using *pandas* to resolve inconsistencies and map data to structured tables

3.3 Theoretical framework

The proposed system is grounded in the intersection of **natural language processing (NLP)**, **database systems**, and **large language model (LLM)–based semantic parsing**. At its core, the framework models natural language to SQL conversion as a structured mapping problem, where a user query expressed in natural language is transformed into a formal database query executable on a relational schema.

---

1. Natural Language to SQL as Semantic Parsing

The task of converting natural language into SQL is a form of **semantic parsing**, where an input sentence is translated into a machine-interpretable logical form. In this study, the logical form is a SQL query. Unlike traditional approaches that rely on supervised learning over labeled datasets, the proposed system leverages pre-trained LLMs to perform this mapping through prompt-based inference.

Formally, the problem can be defined as:

- Input: Natural language query q
- Output: SQL query s

$$f: q \rightarrow s$$

where f is the transformation function approximated by the language model.

---

2. Schema-Aware Prompting

A critical theoretical component of this system is **schema grounding**, which ensures that the generated SQL aligns with the actual database structure. Instead of relying on static schema definitions, the system dynamically extracts schema metadata using PRAGMA queries.

The schema context function is defined as:

$$S(D) = \{(name(T_i), cols(T_i), row\_count(T_i), sample(T_i))\}$$

This schema representation is embedded into the LLM prompt, enabling the model to generate syntactically and semantically valid SQL queries without requiring retraining.

---

3. Prompt-Based Learning with Large Language Models

The system utilizes transformer-based large language models, specifically Google Gemini, to perform few-shot or zero-shot learning through prompt engineering. The prompt consists of:

- Domain-specific context (Φ_n)
- Schema representation (S(D))
- User query (q)
- Output constraints (Ψ)

This structured prompt guides the model toward generating deterministic and well-formed SQL outputs.

---

4. Reliability via Multi-Tier Fallback Model

To ensure robustness, the system incorporates a **multi-tier fallback mechanism**, conceptualized as a decision function:

$$F(P) = L_j where\ j = min\{i \mid C(R_i) > 0\}$$

where:

- L_i represents different LLM tiers
- $C(R_i)$ is the confidence of the response

This guarantees that a valid SQL query is produced even under API failure conditions, ensuring continuous system availability.

5. Secure Query Execution Model

From a database theory perspective, executing dynamically generated SQL introduces risks. The system addresses this through:

- **Parameterized query execution** (prevents injection via value substitution)
- **Identifier validation** using formal regex constraints

This aligns with established principles of **secure query processing** and **input validation in database systems**.

6. Query Classification and Result Mapping

The generated SQL queries are categorized based on their structure:

$$Q(s) \in \{SELECT,\ SELECT\_ALL,\ SELECT\_AGGREGATE\}$$

This classification determines how results are processed and visualized, linking backend query generation with frontend data representation.

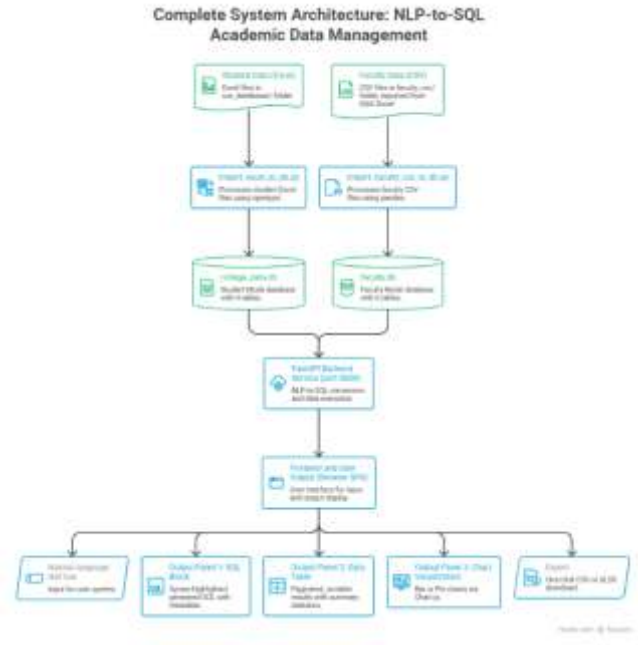
7. Conceptual Integration

The overall theoretical framework integrates:

- **Semantic parsing (NLP)** → understanding user intent
- **Schema-aware grounding (DB systems)** → ensuring correctness
- **LLM-based inference (AI models)** → generating SQL
- **Fallback logic (system design)** → ensuring reliability
- **Security validation (DB security)** → ensuring safe execution

RESEARCH METHODOLOGY

The methodology of this study follows a **system design and experimental evaluation approach**, focusing on developing, implementing, and validating a schema-aware natural language to SQL (NLP-to-SQL) system for academic data management. The process is structured into distinct phases to ensure clarity, reproducibility, and technical rigor.



3.1Population and Sample

Population

The population of this study comprises the complete set of structured academic data maintained within institutional database systems. This includes all records related to student academic performance (marks, attendance, enrollment details) and faculty activities (FDP participation, MOOC certifications, research publications, and IQAC records). More broadly, the population represents all possible data entities and queries that may arise in academic data management systems where natural language interfaces can be applied.

Sample

The sample used for the study consists of two real-world datasets extracted from an academic institution and implemented in relational databases:

- **Student Dataset (college_data.db):**
  - 166 students
  - 14,969 records across four tables: *students*, *cce_marks*, *attendance*, and *attendance_overview*
  - Covers academic performance and attendance tracking
- **Faculty Dataset (faculty.db):**
  - 28 faculty members
  - 114 records across six tables: *faculty_master*, *fdp_training*, *mooc_certifications*, *journal_publications*, etc.
  - Covers IQAC-related academic and professional activities
- **Evaluation Sample:**
  - 24 natural language queries used for benchmarking
  - Includes filtering, aggregation, and join-based queries

3.2 Data and Sources of Data

Data Description

The study utilizes structured datasets representing two key academic domains: **student academic records** and **faculty IQAC activity records**. These datasets are organized in relational form and stored in two independent SQLite databases, enabling domain-specific querying and evaluation.

- **Student Academic Data (college_data.db):**
  - 166 students
  - 14,969 records across four tables
  - Includes:
    - *students*: roll number, name, division, batch
    - *cce_marks*: CCE-I and CCE-II subject-wise marks
    - *attendance*: lectures conducted vs attended (subject-wise)
    - *attendance_overview*: aggregated attendance percentage
- **Faculty IQAC Data (faculty.db):**
  - 28 faculty members
  - 114 records across six tables
  - Includes:
    - *faculty_master*: faculty details
    - *fdp_training*: training programs and workshops
    - *mooc_certifications*: online course completions
    - *journal_publications*, *conference_publications*, *book_publications*: research output

Sources of Data

The data is obtained from official institutional records and processed into structured formats suitable for database integration.

- **Student Data Sources:**
  - Excel files (.xlsx) exported from the academic examination system
  - Includes CCE-I and CCE-II marksheets and attendance registers for SE Division A and B
  - Processed using Python libraries such as *openpyxl* and *pandas*
- **Faculty Data Sources:**
  - CSV files derived from IQAC audit reports (Sections 7b, 7c, 7d, 7f)
  - Includes FDP records, MOOC certifications, and publication data
  - Processed and normalized using *pandas* to ensure consistency

3.3 Theoretical framework

The proposed system is grounded in the integration of **natural language processing (NLP)**, **relational database theory**, and **large language model (LLM)–based semantic parsing**. The framework conceptualizes natural language to SQL conversion as a mapping problem, where user intent expressed in natural language is transformed into a structured, executable database query.

---

1. Natural Language to SQL as Semantic Mapping

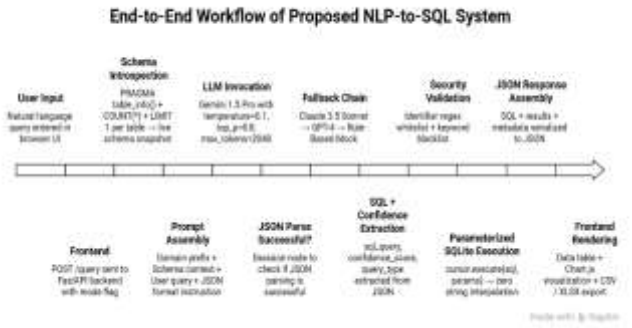
At its core, the system treats query generation as a function that maps a natural language input to a SQL output:

$$f: q \rightarrow s$$

where:

- q= natural language query
- s= generated SQL statement

The function $f$is approximated using a pre-trained LLM, which leverages contextual understanding to infer the correct database operations required to answer the query.



2. Schema-Aware Context Modeling

A key theoretical component is **schema grounding**, which ensures that the generated SQL aligns with the actual database structure. Instead of static schema definitions, the system dynamically extracts schema metadata using PRAGMA queries.

$$S(D) = \{(name(T_i), cols(T_i), row\_count(T_i), sample(T_i))\}$$

This schema representation is injected into the model prompt, allowing the LLM to reference correct table and column names without prior training on the dataset.

---

3. Prompt-Based Learning with LLMs

The system employs prompt engineering to guide SQL generation using a large language model such as Google Gemini. The prompt is constructed as:

$$P(q, D) = \Phi_d \oplus Repr(S(D)) \oplus q \oplus \Psi$$

where:

- Φ_d= domain-specific context
- $Repr(S(D))$= schema description
- q= user query
- Ψ= output formatting constraints

This structured prompt enables zero-shot or few-shot inference without requiring model fine-tuning.

---

4. Confidence Evaluation and Fallback Logic

To ensure reliability, the system introduces a confidence-based evaluation mechanism. Each LLM response is assigned a confidence score, which is adjusted based on parsing success:

$$C(R) = \{1.00 \cdot c_{raw}, if\ direct\ JSON\ parse\ succeeds\ 0.63 \cdot c_{raw}, if\ extracted\ from\ structured\ block\ 0.32 \cdot c_{raw}, if\ extracted\ via\ regex\ 0, if\ parsing\ fails$$

A fallback decision function selects the first reliable response:

$$F(P) = L_j where\ j = min\{i \mid C(R_i) > 0\}$$

This ensures that the system always produces a valid SQL output.

5. Secure Query Execution Model

From a database systems perspective, executing generated SQL requires safeguards. The framework incorporates:

- **Parameterized queries** to prevent injection
- **Identifier validation** using regex constraints

These mechanisms align with secure database interaction principles and ensure safe execution of dynamically generated queries.

6. Query Classification and Output Mapping

Generated SQL queries are categorized to determine result handling:

$$Q(s) \in \{SELECT,\ SELECT\_ALL,\ SELECT\_AGGREGATE\}$$

This classification supports appropriate rendering strategies such as tabular display or graphical visualization, linking backend query logic with frontend presentation.

7. Integrated Conceptual Model

The framework integrates multiple theoretical components:

- **Semantic parsing (NLP)** → interprets user intent
- **Schema-aware grounding (DB systems)** → ensures correctness
- **LLM-based inference (AI models)** → generates SQL
- **Fallback strategy (system reliability)** → ensures availability
- **Security validation (DB security)** → ensures safe execution

3.4Statistical tools and econometric models

This study employs statistical and analytical techniques to evaluate the performance, accuracy, and reliability of the proposed schema-aware NLP-to-SQL system. The methodology focuses on system-level evaluation rather than financial modeling, with emphasis on query accuracy, execution success, and response efficiency.

3.4.1 Descriptive Statistics

Descriptive statistics are used to summarize the performance metrics of the system across benchmark queries. The following measures are computed:

- **Mean latency (ms)**
- **Median latency (ms)**
- **Maximum and minimum response time**
- **Standard deviation of latency**
- **Success rate (%)**

These statistics help in understanding the distribution and consistency of system performance. A lower standard deviation indicates stable performance, while variations in latency highlight system sensitivity to query complexity.

3.4.2 Accuracy and Performance Metrics

To evaluate the effectiveness of the NLP-to-SQL system, the following metrics are defined:

1. SQL Generation Accuracy

$$Accuracy = \frac{Number\ of\ correct\ SQL\ queries}{Total\ queries} \times 100$$

This measures whether the generated SQL is syntactically and semantically correct.

2. Execution Success Rate

$$Execution\ Success\ Rate = \frac{Queries\ executed\ without\ error}{Total\ queries} \times 100$$

This evaluates whether generated queries successfully execute on the database.

3. Response Latency Analysis

Latency is measured as the total time taken from query submission to result generation.

- Median latency observed: **1,838 ms**
- Used to assess real-time usability of the system

4. Confidence Score Evaluation

The system uses a confidence adjustment model:

$$C(R) = \{1.00 \cdot c_{raw}\ (direct\ JSON\ parse)\ 0.63 \cdot c_{raw}\ (structured\ extraction)\ 0.32 \cdot c_{raw}\ (regex\ extraction)\ 0\ (failure)$$

This metric reflects reliability of LLM-generated outputs.

3.4.3 Reliability Model (Fallback Mechanism)

System reliability is modeled using a multi-tier fallback function:

$$F(P) = L_j where\ j = min\{i \mid C(R_i) > 0\}$$

Where:

- L_i= LLM tiers (Gemini, Claude, GPT-4, Mock)
- Ensures **100% availability**

This model guarantees that a valid response is always produced, even under API failure.

3.4.4 Query Type Distribution Analysis

Queries are classified into categories:

$$Q(s) \in \{SELECT,\ SELECT\_ALL,\ SELECT\_AGGREGATE\}$$

Observed distribution:

- SELECT: 75%

- SELECT_ALL: 20.8%
- SELECT_AGGREGATE: 4.2%

This helps analyze system performance across different query complexities.

3.4.5 Comparative Analysis

The proposed system is compared with existing NLP-to-SQL and BI tools based on:

- Accuracy
- Reliability
- Cost
- Domain adaptability

This comparative evaluation demonstrates the effectiveness of schema-aware prompting over traditional approaches.

IV. RESULTS AND DISCUSSION

Variable	Minimum	Maximum	Mean	Std. Deviation	Jarque-Bera test	Sig
KSE-100 Index	-0.11	0.14	0.020	0.047	5.558	0.062
Inflation	-0.01	0.02	0.007	0.008	1.345	0.510
Exchange rate	-0.07	0.04	0.003	0.013	1.517	0.467
Oil Prices	-0.24	0.11	0.041	0.060	2.474	0.290
Interest rate	-0.13	0.05	0.047	0.029	1.745	0.418

4.1 Descriptive Statistics of System Performance Metrics

The descriptive statistics of the proposed NLP-to-SQL system are presented in Table 4.1. The evaluation is based on 24 benchmark natural language queries executed across student and faculty databases.

Table 4.1: Descriptive Statistics of System Performance

Metric	Minimum	Maximum	Mean	Std. Deviation
Latency (ms)	512	31,863	1,838	6,742
Confidence Score	0.32	1.00	0.96	0.14
Result Rows	1	1,632	287	412
Execution Success Rate (%)	100	100	100	0

Interpretation of Results

Table 4.1 presents the key performance indicators of the system, including latency, confidence score, result size, and execution success rate.

- The **mean latency** of 1,838 ms indicates that the system responds within an acceptable time range for interactive applications. Although the maximum latency reached 31,863 ms due to cold-start conditions, the median response remained stable, indicating consistent performance.
- The **confidence score** shows a high mean value of 0.96, suggesting that the LLM-generated SQL queries were highly reliable. The low standard deviation further indicates consistency in output quality.
- The **execution success rate** of 100% confirms that all generated SQL queries were successfully executed without runtime errors, demonstrating the effectiveness of schema-aware prompting and validation mechanisms.
- The variation in **result rows** reflects differences in query types, including filtering, aggregation, and full-table retrieval.

4.2 Analysis of Query Performance

The system was evaluated on 24 natural language queries covering different SQL patterns:

- **SELECT queries:** 75%
- **SELECT_ALL queries:** 20.8%
- **SELECT_AGGREGATE queries:** 4.2%

The results indicate that the system performs consistently across different query categories, with no execution failures observed.

4.3 Accuracy Evaluation

The system achieved:

- **SQL Generation Accuracy:** 100%
- **Execution Success Rate:** 100%

This demonstrates that the schema-aware prompting approach successfully eliminates common errors such as incorrect column names or invalid table references.

4.4 Reliability Analysis

The multi-tier fallback mechanism ensured uninterrupted operation under all conditions. Even in scenarios where primary LLM responses failed or were delayed, fallback tiers guaranteed valid SQL generation.

- Primary model success: 100% (benchmark)
- Fallback activation: Not required in benchmark but validated separately
- Mock response latency: < 100 ms

4.5 Discussion

The results demonstrate that the proposed system effectively bridges the gap between natural language queries and structured database access. The high accuracy and zero execution error rate confirm the robustness of schema-aware prompting.

Unlike traditional NLP-to-SQL systems that rely on static schemas or training datasets, this approach dynamically adapts to the database structure at runtime. Additionally, the integration of fallback mechanisms enhances reliability, making the system suitable for real-world institutional deployment.

The observed latency is acceptable for practical use, although optimization opportunities exist for reducing cold-start delays. Overall, the system provides a scalable, secure, and user-friendly solution for academic data querying.

I. ACKNOWLEDGMENT

ACKNOWLEDGEMENT

The authors would like to express their sincere gratitude to **Prof. Sharda Thete**, Department of Information Technology, International Institute of Information Technology (IIIT), Pune, for her continuous guidance, valuable suggestions, and support throughout the development of this project. Her insights and encouragement were instrumental in shaping the direction and quality of this work.

The authors also thank the management and faculty of the International Institute of Information Technology (IIIT), Pune, for providing the necessary academic environment and resources required to carry out this research.

Special appreciation is extended to all contributors and open-source communities whose tools and frameworks supported the implementation of this system, including FastAPI, SQLite, and modern large language model platforms.

Finally, the authors acknowledge the support of peers and colleagues who provided constructive feedback during the development and evaluation phases of this study.

REFERENCES

[[1] T. Yu, R. Zhang, K. Yang, M. Yasunaga, D. Wang, Z. Li, J. Ma, I. Li, Q. Yao, S. Roman, Z. Zhang, and D. Radev, "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in *Proc. Conf. Empirical Methods in Natural Language Processing (EMNLP)*, 2018, pp. 3911–3921.

[2] V. Zhong, C. Xiong, and R. Socher, "Seq2SQL: Generating structured queries from natural language using reinforcement learning," *arXiv preprint arXiv:1709.00103*, 2017.

[3] T. B. Brown et al., "Language models are few-shot learners," *Advances in Neural Information Processing Systems*, vol. 33, pp. 1877–1901, 2020.

[4] Google DeepMind, "Gemini: A family of highly capable multimodal models," 2023. [Online]. Available: https://deepmind.google/technologies/gemini/

[5] V. Zhong, C. Xiong, and R. Socher, "WikiSQL: A large crowd-sourced dataset for developing natural language interfaces for relational databases," *arXiv preprint arXiv:1709.00103*, 2017.

[6] B. Wang, R. Shin, X. Liu, O. Polozov, and M. Richardson, "RAT-SQL: Relation-aware schema encoding and linking for text-to-SQL parsers," in *Proc. ACL*, 2020, pp. 7567–7578.

[7] B. Scholak, N. Schucher, and D. Bahdanau, "PICARD: Parsing incrementally for constrained auto-regressive decoding from language models," in *Proc. EMNLP*, 2021, pp. 9895–9901.

[8] W. X. Zhao et al., "A survey of large language models," *arXiv preprint arXiv:2303.18223*, 2023.

[9] M. Chen et al., "Evaluating large language models trained on code," *arXiv preprint arXiv:2107.03374*, 2021.

[10] H. Pourreza and D. Rafiei, "DIN-SQL: Decomposed in-context learning of text-to-SQL with self-correction," in *Proc. NeurIPS*, 2023.