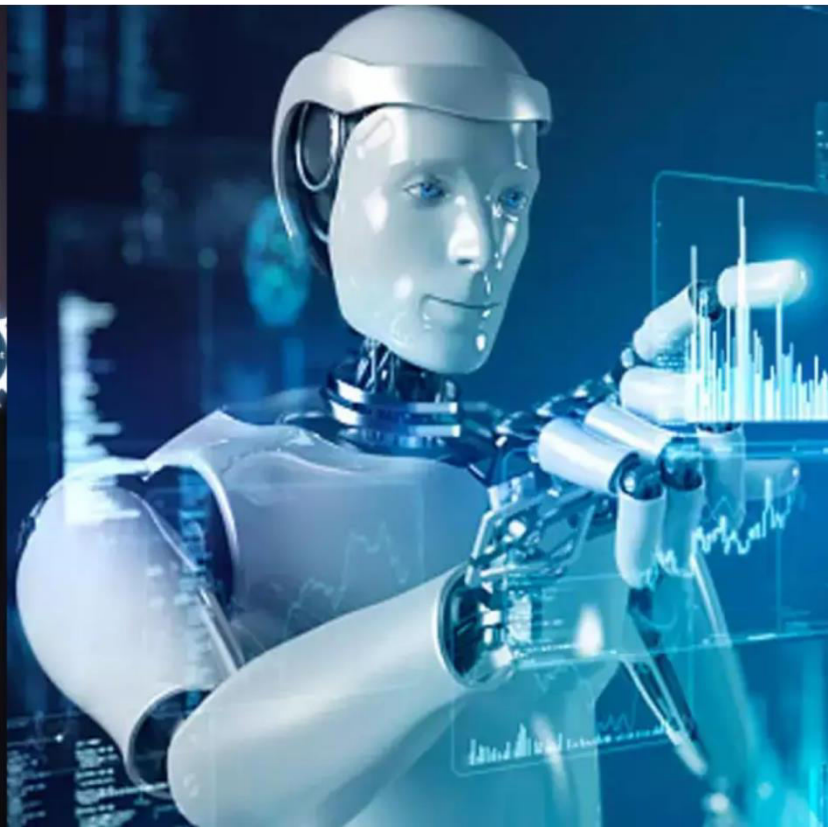
IJIRSET
International Journal of Innovative Research in Science | Engineering | Technology

ISSN(O): 2319-8753
ISSN(P): 2347-6710

International Journal of Innovative Research in Science Engineering and Technology (IJIRSET)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.935

Volume 15, Issue 5, May 2026

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 5, May 2026

|DOI: 10.15680/IJIRSET.2026.1505169|

A Hybrid Schema-Aware Framework for Conversational Text-to-SQL Generation with Deterministic Query Synthesis

Veda Bhadane[1], Prajakta Patil[2], Anshuli Patil[3], Dhanashree Patil[4], Dr. Sarika S Jadhav[5], Prof. Aeman Patel[6]

U.G. Student, Department of Computer Science, Padmabhooshan Vasantdada Patil Institute of Technology, Bavdhan, Pune (MH), India

Professor, Department of Computer Science, Padmabhooshan Vasantdada Patil Institute of Technology, Bavdhan, Pune (MH), India

ABSTRACT: The purpose of a Natural Language to SQL (NL2SQL) system is to allow users to query relational databases using their natural language instead of using SQL. Current approaches are usually based on one of two paradigms: rule-based approaches that are not flexible, and approaches based on Large Language Models (LLM) that have hallucinations, are not interpretable, and are risky. In this paper, we propose a new framework that is based on a hybrid model and is schema aware. In this framework, we propose a new methodology that is based on dividing the NL2SQL problem into structured layers. In our proposed methodology, we use an LLM for the semantic interpretation and entity recognition. Moreover, we use deterministic query construction and validation. In this framework, we use an automatic schema awareness and mapping. This framework is general and can be used with any schema in the database[15].

KEYWORDS: Text-to-SQL, Natural Language Interfaces to Databases (NLIDB), Conversational Query Systems, Schema Linking, Intermediate Representation (IR), Deterministic Query Generation, Large Language Models (LLMs), Query Validation, Database Query Optimization, Human-Computer Interaction

I. INTRODUCTION

Relational databases are typically used by modern organizations to store vast amounts of data. SQL knowledge is typically required to access this type of data. However, many users lack the technical expertise required to write SQL queries, including business analyst managers and subject matter experts. Because of these restrictions, they frequently depend on technical teams or database administrators to obtain the necessary data, which ultimately slows down the decision-making process [1].

Due to strict mappings and little linguistic flexibility, traditional rule-based systems produce deterministic and comprehensible results but are unable to generalize across domains. However, natural human language can now be used to communicate with computer systems thanks to recent developments in large language models and natural language processing. However, it also brings with it problems like unsafe query generation, lack of control, and hallucinated schema elements [7][15].

A methodology that eliminates the shortcomings of both approaches while combining their advantages is obviously needed. This paper presents a hybrid framework that combines schema-aware deterministic SQL generation and validation mechanisms with LLM-based semantic understanding. The objective is to create a system that is safe, interpretable, adaptive to various database schemas, and flexible [4][5].

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 5, May 2026

|DOI: 10.15680/IJIRSET.2026.1505169|

II. RELATED WORK

Natural Language to SQL, which is also known as Text-to-SQL has changed a lot over time. It used to be based on rules. Now it uses deep learning and big language models. People who study this can break it down into three parts: the neural models, the methods that know, about the structure and schema and the new systems that use big language models or Natural Language to SQL systems that use large language models [15].

2.1 Early Neural Text-to-SQL Models

The first neural models, such as Seq2SQL and SQLNet used a sequence-to-sequence method to create SQL queries from language. These models showed that it is possible to learn SQL generation from data. They had trouble with complicated queries that involved joining tables nested structures and generalizing to different database schemas [1][2]. Later models like Syntax SQLNet did better by using SQL grammar rules to reduce mistakes during query generation [3].
However these models still had issues with handling schemas and complex queries that involved multiple tables.

2.2 Schema-Aware and Structured Methods

To improve schema generalization models like RAT-SQL used an attention mechanism to understand relationships between tables and columns. This helped RAT-SQL perform well on benchmarks like Spider that test models across domains [4].
More recent models, such as RESDSQL separate the process of linking the schema to SQL generation. This allows them to better match phrases in language to database structures [5].
The DBCopilot model also proposes a pipeline that breaks down the process into schema routing, generating candidates and synthesizing SQL [6].
These models show that having structured steps and grounding in the schema is important, for making models more robust and understandable.

2.3 LLM-Based Text-to-SQL Systems

The big language models that are available now have really improved how well Text-to-SQL works. For example systems like SQL-PaLM use these models that have already been trained to learn from the context and make SQL queries that are very accurate [7].
Other work like CodeS and DAIL-SQL shows that things like designing the prompts putting the schema in order and picking the right examples are very important for how well the system works [8][9].
Even though these systems do well on tests they often need big prompts to work do not check things carefully enough and can make queries that are not safe or that are just made up which limits how they can be used in the real world [8][9].

2.4 Conversational Text-to-SQL

There are datasets like CoSQL. Sparc that look at conversations where people talk back and forth and the system has to remember what was said before [12][13].
These studies show that there are challenges like keeping track of what is happening figuring out what people mean when they are not clear and making queries better which are all necessary, for interfaces that let people talk to databases [12][13].

2.5 Limitations of Existing Approaches

The methods we have now work well on some datasets like Spider and BIRD. However when we use these methods in the world we find that things are a lot more complicated. For example the BIRD Benchmark and Spider 2.0 tests show that these models have trouble with schemas, ambiguous queries and constraints that happen in real life [10].
In addition systems that use Large Language Models often do not have: [7][8][9][15]

- control over how SQL's generated that is consistent and predictable
- good ways to check if the results are correct
- guarantees that the database will be safe when we execute the queries

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 5, May 2026

|DOI: 10.15680/IJIRSET.2026.1505169|

2.6 Positioning of Proposed Work

The system we are proposing is different from other systems that only use Large Language Models or only use neural approaches. Our system uses a combination of understanding the meaning of the data and generating SQL in a way that's consistent and predictable [1][2][3]

The main things that make our work different are:

- mapping entities to the schema in a way that is based on the schema
- representing queries in a way before generating the SQL
- using rules to generate the SQL
- checking the results for errors before executing the queries

This design helps us get more accurate results makes it easier to understand what is happening and makes the system safer. This makes it more suitable for use, in database systems that're small to medium in size.

III. PROBLEM STATEMENT

Data stored in databases is really important for making decisions in modern organisations. However to get to this data you need to know Structured Query Language, which a lot of people who are not tech savvy do not know. Because of this people have to rely on staff to get the information they need and this causes delays people do not get as much work done and they cannot explore the data as much as they want to [1].

The tools we have now to access data like dashboards and reporting tools can only do what they were made to do. Cannot handle requests that are not planned for. Also the old way of using language to talk to databases often has trouble understanding complicated requests dealing with big database plans and making correct SQL statements [1][2]. Now that we have Large Language Models we can make systems that understand what people are asking for in language and turn that into SQL statements that can be used. We still have problems like understanding the database plan making sure the query is correct figuring out what the person really means and making sure the query is safe to run [7][8].

So we need to make a system that is strong and can handle a lot of users and lets people talk to databases using natural language while making sure the SQL statements are correct the query is safe and the data is retrieved quickly. The LLM Assisted Conversational Text-to-SQL system we are proposing is trying to solve these problems by giving people a smart way to access data.

The LLM Assisted Conversational Text-to-SQL system is going to make it easy for people to use language to get data, from databases and it will make sure that the SQL statements are accurate and the data is retrieved safely and quickly. This will help people who're not tech savvy to use databases easily and it will make it easier for organisations to make decisions based on the data they have.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

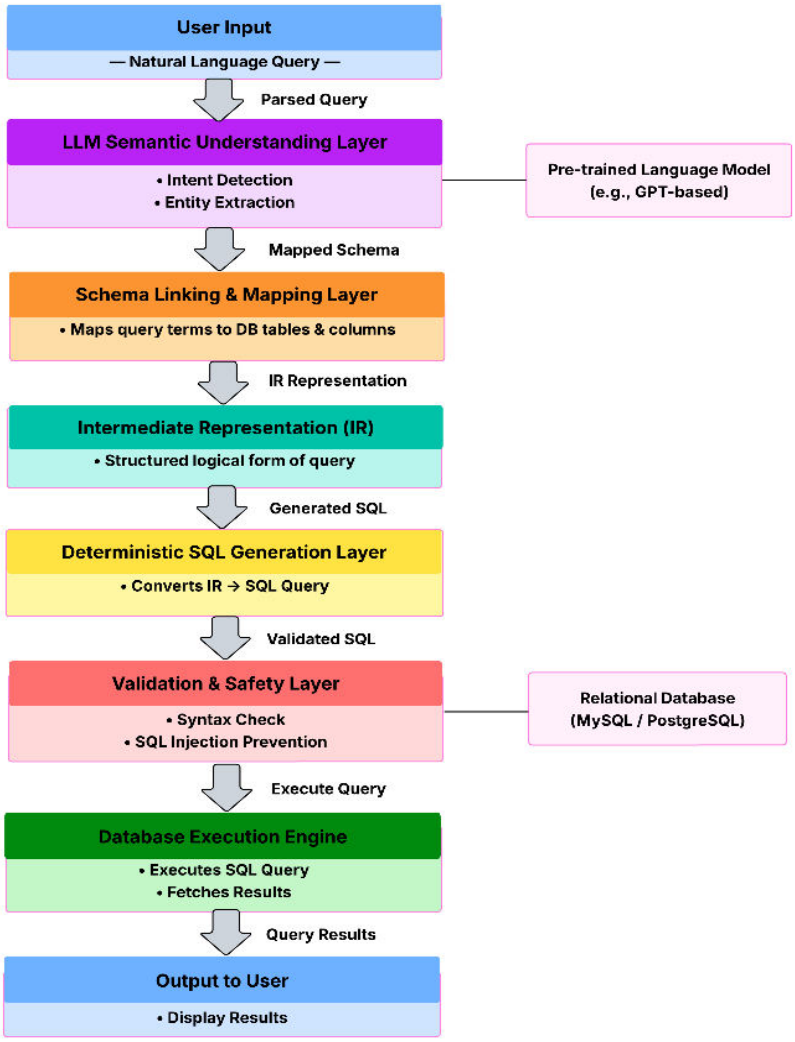
Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 5, May 2026

|DOI: 10.15680/IJIRSET.2026.1505169|

IV. SYSTEM ARCHITECTURE



The proposed architecture uses a step-by-step process to turn a language query into a SQL statement that the computer can understand. Each step has a job making sure the process is controlled less confusing and more reliable. The design uses both a language model and strict rules to balance flexibility and accuracy.

4.1 User Input Layer

The process starts when the user gives a query in language, which is an informal request for data. This query can be simple or complex like asking for data or analyzing something. The system takes this input as it is and gets it ready to understand what it means without needing the user to know SQL or database details.

4.2 LLM Semantic Understanding Layer

In this step a pre-trained language model helps understand what the query means. The model figures out what the user wants and picks out details, like attributes, conditions and numbers. This stage helps the system turn language into a formal understanding that captures what the query needs. By using the language models abilities the system gets better at understanding ways of saying things [7][8].

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 5, May 2026

|DOI: 10.15680/IJIRSET.2026.1505169|

4.3 Schema Linking and Mapping Layer

The semantic elements are then connected to the database schema that matches. This part helps to link what the user says to how the database's actually set up. It does this by connecting the things we found to the names of the tables and columns. This helps to stop the system from making things up and makes the query generation process more accurate [4][5].

4.4 Intermediate Representation Layer

The database schema that we connected is then changed into a form. This form shows the structure of the query. It does not use the SQL syntax but instead shows what the query is trying to do how the data is related and what needs to be done. This helps to make the query easier to understand. Gives us a controlled way to generate the query [5][6].

4.5 Deterministic SQL Generation Layer

The middle form is then changed into a SQL query. We use a set of rules to do this. This part of the process creates a query by building the parts, such as the parts that say what to select, where to get it from what conditions to apply how to group things and how to order things. The semantic elements and database schema and SQL query are all important here. This process is different, from the way a large language model generates queries [8][9].

4.6 Validation and Safety Layer

Before the system runs the SQL query it checks the query to make sure it is correct and safe. The system checks the syntax of the SQL query it checks the schema and it makes sure that nobody can do anything like inject bad SQL code or make unauthorized changes. This step is, like a safety net that makes sure good and safe SQL queries get to the part of the system that runs them.

4.7 Database Execution Engine

When the SQL query is okay the system runs it on the database. The part of the system that runs the query does it quickly. Gets the data that is needed. This part of the system talks directly to the database. Makes sure the query runs correctly and quickly.

4.8 Output Layer

The system then shows the user the results it got from the database in a way that's easy to read. The system may also show the user the SQL query it used. The user can see what happened. This is the step and it gives the user the results and helps the user understand how the SQL query worked. The system shows the user the SQL query. This helps the user see what the system did with the SQL query.

V. PROPOSED METHODOLOGY

The proposed methodology follows a structured pipeline that converts natural language queries into executable SQL statements through a combination of semantic interpretation, schema grounding, and rule-based synthesis. Unlike purely generative approaches, the system enforces controlled transformations at each stage to ensure accuracy, interpretability, and safety [4][5].

5.1 Natural Language Interpretation

The methodology begins by interpreting the user's natural language query using a semantic understanding model. The objective at this stage is to extract the underlying intent and identify key elements such as attributes, conditions, and operations. Instead of directly generating SQL, the system focuses on building a structured understanding of the query, allowing it to handle variations in phrasing and linguistic ambiguity more effectively.

5.2 Intent Identification and Query Decomposition

Following interpretation, the query is decomposed into its logical components based on the identified intent. This involves determining whether the query requires selection, filtering, aggregation, or ordering operations. By explicitly decomposing the query into smaller functional parts, the system reduces complexity and enables more controlled construction of SQL statements.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 5, May 2026

|DOI: 10.15680/IJIRSET.2026.1505169|

5.3 Entity Recognition and Extraction

The system then extracts relevant entities from the query, including potential table names, column references, conditional values, and numerical constraints. These entities form the core building blocks required for SQL generation. The extraction process ensures that all meaningful components of the query are captured before mapping them to the database schema [4][5].

5.4 Schema Grounding and Alignment

In this stage, the extracted entities are aligned with the actual database schema. Since user queries often contain informal or ambiguous terminology, the system applies mapping rules and schema metadata to resolve these terms into valid table and column names. This grounding process ensures that all subsequent operations are based on valid database elements, significantly reducing the likelihood of hallucination or incorrect references [11].

5.5 Intermediate Representation Construction

Once schema alignment is completed, the system constructs an intermediate representation that captures the logical structure of the query. This representation includes selected attributes, filtering conditions, join relationships, and aggregation requirements in a structured format. By decoupling logical representation from SQL syntax, the system improves transparency and allows for systematic query construction [5].

5.6 Deterministic SQL Synthesis

The structured intermediate representation is then translated into a SQL query using deterministic rules. Each component of the representation is mapped to its corresponding SQL clause, ensuring that the final query adheres to proper syntax and logical consistency. This approach eliminates the unpredictability of direct SQL generation and provides greater control over query construction [1][2].

5.7 Query Validation and Safety Enforcement

Before execution, the generated SQL query undergoes a validation process to ensure correctness and security. The system checks for syntactic validity, verifies schema consistency, and restricts unsafe operations. This step ensures that only reliable and secure queries are executed, making the system suitable for practical deployment [9].

5.8 Query Execution and Result Retrieval

After validation, the SQL query is executed on the database, and the results are retrieved. The system ensures efficient interaction with the database and handles execution errors gracefully. The retrieved results are then passed to the output stage for presentation.

VI. EXPERIMENTAL RESULTS AND DISCUSSION

The performance of the proposed system was evaluated using a set of natural language queries designed to cover a variety of SQL operations. The results demonstrate that the hybrid architecture provides a strong balance between accuracy, robustness, and safety. To present the evaluation clearly, quantitative results are summarized in Table 1 [9][11].

Table 1: Overall System Performance

Metric	Value
Exact Match Accuracy	78%
Execution Accuracy	86%
Invalid Query Rate	6%
Hallucination Rate	4%
Average Response Time	1.2 sec

The results show that while exact match accuracy is moderate, execution accuracy is significantly higher, indicating that the system is able to produce correct outputs even when the SQL structure varies. The relatively low invalid query

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 5, May 2026

|DOI: 10.15680/IJIRSET.2026.1505169|

and hallucination rates highlight the effectiveness of the validation and schema mapping components in ensuring reliability [9][10]

To further understand system behaviour, performance was analyzed across different query types, as shown in Table 2.

Table 2: Accuracy by Query Type

Query Type	Accuracy
Simple SELECT	95%
WHERE Filters	90%
JOIN Operations	80%
GROUP BY / Aggregation	72%
ORDER BY	85%

The system performs strongly on simpler queries, where direct mappings between natural language and schema elements exist. However, performance decreases for more complex operations such as joins and aggregations, which require deeper relational reasoning and more precise interpretation of intent. To evaluate the effectiveness of the hybrid approach, the system was compared with baseline methods, as shown in Table 3 [10].

Table 3: Comparison with Baseline Approaches

Approach	Accuracy	Invalid Query Rate	Hallucination Rate
Rule-Based Only	60%	3%	1%
LLM Only	72%	12%	10%
Proposed Hybrid System	86%	6%	4%

The comparison shows that rule-based systems are reliable but lack flexibility, while LLM-only approaches provide better language understanding but suffer from higher error rates. The proposed hybrid system achieves a balance by combining semantic understanding with deterministic control, resulting in improved overall performance [1][2][3].
To provide deeper insight into system limitations, selected failure cases are presented in Table 4.

Table 4: Sample Error Analysis

Input Query	Generated SQL	Issue
"Top 5 customers"	SELECT * FROM customers LIMIT 5	Missing ranking criteria
"Total sales by city"	Incorrect GROUP BY usage	Aggregation error
"Customers with highest orders"	Missing JOIN condition	Join misinterpretation

These examples highlight that most errors arise from ambiguity in user queries or limitations in handling complex relational structures [10].

Overall, the results demonstrate that the proposed system improves execution accuracy and reduces hallucination compared to baseline approaches, while maintaining efficient response times. However, challenges remain in handling ambiguous queries and complex SQL constructs, suggesting directions for future improvement [9][10].

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 5, May 2026

|DOI: 10.15680/IJIRSET.2026.1505169|

VII. ADVANTAGES

1.Ease of Use-Instead of writing complicated SQL queries, users are able to communicate with the database in plain language, making the system simple and convenient to use for everyone.
2.Elimination of SQL Dependency-Even without knowing SQL, users can easily interact with the database because the system converts their plain English text into proper SQL queries on its own.
3.Improved Efficiency-It automatically translates natural language into SQL statements, making data retrieval faster and more efficient.
4.Redused Syntax and Logical Error-Errors that usually occur while writing SQL manually can be reduced, as the system handles query generation automatically.
5.Automation of Query Generation-The conversion of natural language into SQL queries is performed automatically, thereby minimizing manual effort.
6.Secure Query Execution-The system checks every query before execution ton make sure it is safe and only used for reading data, so that no damage can be caused to database.
7.Modular Design-The step-by-step approach is followed in the system, where tasks like natural language processing, query generation and validation are handled separately for debugging and maintaining the system.
8.Schema Query Generation –By using schema mapping, the database structure is taken into account, allowing system to create queries that are both accurate and relevant.

VIII. FUTURE SCOPE

1.Advance Large Language Models (LLMs)-In future, the advanced language models can enhance the system to interpret the text more accurately and process complex and ambiguous queries [7][8].
2.Multi-Language Support-The System may support multiple languages in future so that more users can access it without language barriers.
3.Sequential Conversational Memory-The system will keep the track of previous queries, so users can continue the conversation without restating everything [12][13].
4.Voice Based Query-Users will be able to interact with the system through voice, where speech is transformed into text using speech recognition techniques.
5.Enhances Security -In future System can be made more secure by adding features like access control, data masking and query tracking, which will help in protecting highly sensitive data and suitable for larger applications.
6.Data Visualization-At present system shows result in table format which can be upgraded to display through graphs, charts and dashboards.
7.Integration with Big Data-The system can be extended to support big data frameworks such as Hadoop, Spark and Cloud Services which will analyze large amounts of data effectively.

IX. CONCLUSION

In this paper, we represented a Conversational SQL Query Generator that allows user to interact with relational databases using natural language. The natural language is then automatically converted into executable SQL statements. The Hybrid approach used in project brings together natural language understanding and schema-based rule to generate queries. This combination helps to maintain balance between accuracy and flexibility also ensuring that the generated queries are understandable and secure.

The system is implemented using Python along with Streamlit and SQLite, which generates solution without the need for heavy infrastructure. The system successfully bridges the gap between non-technical users and database access.
This project reflects the growing role of AI in improving use of database. With further improvements and extensions, the system can be expanded for more robust solutions that can be applied across different applications and domains.

International Journal of Innovative Research of Science, Engineering and Technology (IJIRSET)

Impact Factor 9.935

|www.ijirset.com |A Monthly, Peer Reviewed & Refereed Journal| e-ISSN: 2319-8753| p-ISSN: 2347-6710|

Volume 15, Issue 5, May 2026

|DOI: 10.15680/IJIRSET.2026.1505169|

REFERENCES

[1] Zhong, V., Xiong, C., & Socher, R. (2017). Seq2SQL: Generating Structured Queries from Natural Language. arXiv:1709.00103.
[2] Xu, X., Liu, C., & Song, D. (2017). SQLNet: Generating Structured Queries from Natural Language without Reinforcement Learning. arXiv:1711.04436.
[3] Yu, T., et al. (2018). SyntaxSQLNet: Syntax Tree Networks for Complex SQL Generation. arXiv:1810.05237.
[4] Wang, B., et al. (2020). RAT-SQL: Relation-Aware Schema Encoding for Text-to-SQL Parsing. arXiv:1911.04942.
[5] Li, H., et al. (2023). RESDSQL: Decoupling Schema Linking and Parsing for Text-to-SQL. arXiv:2302.05965.
[6] Zhang, Y., et al. (2023). DBCopilot: A Framework for Text-to-SQL with Schema Routing. arXiv:2312.03463.
[7] Sun, R., et al. (2023). SQL-PaLM: Improved Large Language Model Adaptation for Text-to-SQL. arXiv:2306.00739.
[8] Li, J., et al. (2024). CodeS: Towards Building Open-source Language Models for Text-to-SQL. arXiv:2402.16347.
[9] Gao, D., et al. (2023). Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. arXiv:2308.15363.
[10] Yu, T., et al. (2018). Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing. arXiv:1809.08887.
[11] Li, X., et al. (2023). BIRD: A Benchmark for Text-to-SQL in Realistic Databases. arXiv:2305.03111.
[12] Yu, T., et al. (2019). CoSQL: A Conversational Text-to-SQL Challenge. ACL.
[13] Yu, T., et al. (2019). SParC: Cross-Domain Semantic Parsing in Context. ACL.
[14] Chang, S., et al. (2024). Spider 2.0: Evaluating Text-to-SQL Systems in Realistic Settings. arXiv.
[15] Zhu, C., et al. (2024). A Survey on Employing Large Language Models for Text-to-SQL Tasks. arXiv.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi cross ref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN SCIENCE | ENGINEERING | TECHNOLOGY

9940 572 462 6381 907 438 ijirset@gmail.com

www.ijirset.com

Scan to save the contact details