**Talk2SQL: AI-Driven Natural Language Database Query and Insights**

**25-FYP-306**



**Session 2022-2026**

**BACHELOR OF SCIENCE**

**IN**

**ARTIFICIAL INTELLIGENCE**

**SUBMITTED BY**

Ayesha Fatima 22-NTU-CS-1339

Muhammad Sheeraz 22-NTU-CS-1356

Sumaiya Shehzadi 22-NTU-CS-1376

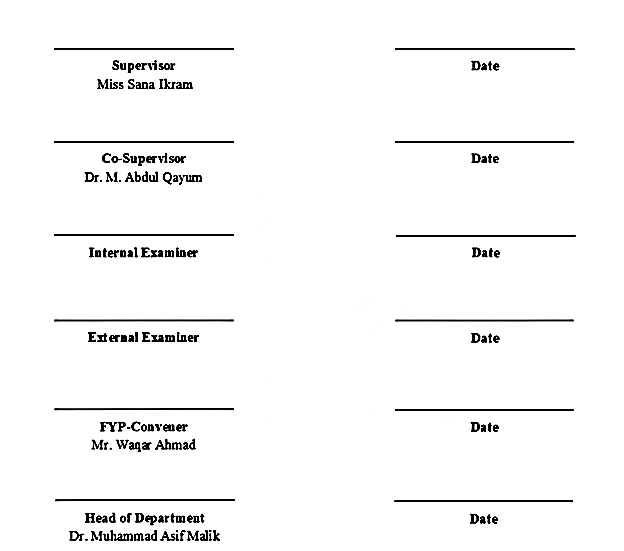
**Supervised by Miss Sana Ikram**

**Co-Supervised by Dr. M. Abdul Qayum**

**DEPARTMENT OF COMPUTER SCIENCE NATIONAL TEXTILE UNIVERSITY, FAISALABAD, PAKISTAN**

# CERTIFICATIONS

This project report written by **Ayesha Fatima (22-NTU-CS-1339), Muhammad Sheeraz (22-NTU-CS-1356)**, and **Sumaiya Shehzadi** **(22-NTU-CS-1376)** under the direction of their supervisors and approved by all the members of the FYP committee, has been presented to and accepted by the Department of Computer Science, in the partial fulfillment of the requirement of the degree of Bachelor of Science in Artificial Intelligence.



# ACKNOWLEDGEMENT

First and foremost, in the name of Allah, the Most Beneficent and Merciful. All praise and appreciation be to Allah. A lot of love for our dear Holy Prophet MUHAMMAD (S.A.W), whose wisdom always leads us along the correct road. Following that, we proudly state that the project's success is dependent on our guardians' unending prayers and support.

We would also like to express our collective gratitude towards our supervisor, **Miss Sana Ikram**, for his assistance and patience during this project, particularly with the report and presentation towards the end. His interest in this project was very beneficial. Every time we meet with him, he is always ready to aid us with amazing ideas, discussions, and suggestions that help us complete our project and documentation. We also wish to thank everyone who assisted us at various phases of the project.

Many thanks to our honored co-supervisor, **Dr. Muhammad Abdul Qayyum**, for his management abilities, which allowed us to divide our job more simply and comprehensively. He took a strong interest in our talents and attempted to polish them.

We cannot miss the assistance of the Department of Computer Science (DCS), NTU services. Special thanks to them for providing us with a location to sit and work in the DCS department, as well as on-demand help. Their helpful attitude was exceptional, and it contributed significantly to the project's overall success.

**Signed By:**

Project ID: FYP-306

# ABSTRACT

Retrieving meaningful data from relational databases typically requires a database administrator to write queries and a business or data analyst to further analyze the retrieved data for predictions and business insights. This multi-step dependency consumes significant time, effort, and cost. While large language models can assist in data analysis, organizations are often reluctant to share confidential and private data over the internet due to concerns related to data privacy, security, and compliance.

Talk2SQL addresses this gap by enabling users to retrieve results and gain insights directly from relational databases using simple English or Roman Urdu through text or voice-based natural language prompts. The system is designed to keep organizational data private and secure, ensuring the protection of PII and sensitive business information. Talk2SQL can be deployed locally or on a self-hosted cloud server and utilizes database schema awareness to generate database-specific queries, avoiding the risks associated with general-purpose LLMs that rely on user data for training.

The platform is delivered as a user-friendly web application that allows users to interact with SQL-based databases such as PostgreSQL, Superbase, and SQL Server using intuitive prompts like “give me yesterday’s sales report” or “which item sold the most last month.” It enforces role-based access control, allowing read-only operations for general users while reserving write operations for administrators. Additionally, Talk2SQL mitigates security risks such as SQL injections by blocking destructive queries and requiring explicit confirmation for sensitive operations, ensuring safe and controlled database interaction.

# TABLE OF CONTENTS

**Description Page #**

[CERTIFICATIONS II](#_Toc221522582)

[ACKNOWLEDGEMENT III](#_Toc221522583)

[ABSTRACT IV](#_Toc221522584)

[TABLE OF CONTENTS V](#_Toc221522585)

[LIST OF FIGURES X](#_Toc221522586)

[LIST OF ABBREVIATIONS XI](#_Toc221522587)

[1.0 Introduction 1](#_Toc221522588)

[1.1 Problem Background 2](#_Toc221522589)

[1.3 Purpose 2](#_Toc221522590)

[1.4 Scope 2](#_Toc221522591)

[1.5 Objectives 3](#_Toc221522592)

[1.6 Research Goals 3](#_Toc221522593)

[1.6.1 Improving Data Accessibility and Decision-Making 3](#_Toc221522594)

[1.6.2 Enhancing Data Privacy and Security 3](#_Toc221522595)

[1.6.3 Advancing Intelligent and Responsible AI Integration 4](#_Toc221522596)

[1.6.4 Reducing Operational Cost and Time Overhead 4](#_Toc221522597)

[1.6.5 Contributing to Global Digital and Sustainable Development Goals 4](#_Toc221522598)

[1.7 Cost Benefit Analysis of Talk2SQL 5](#_Toc221522599)

[1.7.1 Cost Comparison with Existing Database AI and Analytics Tools 6](#_Toc221522600)

[1.7.3 Affordability and Scalability 7](#_Toc221522601)

[1.8 Proposed Solution 7](#_Toc221522602)

[1.8.1 AI-Powered Natural Language Interface for Users 7](#_Toc221522603)

[1.8.2 Secure Backend System 8](#_Toc221522604)

[1.8.3 Result Interpretation and Visualization 8](#_Toc221522605)

[1.8.4 Role-Based User Management 8](#_Toc221522606)

[1.8.5 Development of AI Models 9](#_Toc221522607)

[1.8.6 User Interface Development 9](#_Toc221522608)

[1.9 Project Scheduling 9](#_Toc221522609)

[1.10 Risk Management 10](#_Toc221522610)

[1.10.1 Algorithm and AI Model Risk 10](#_Toc221522611)

[1.10.2 Security and Privacy Risks 10](#_Toc221522612)

[1.10.3 User Interface and Experience Risk 10](#_Toc221522613)

[1.10.4 Integration and Compatibility Risks 11](#_Toc221522614)

[1.10.5 Performance and Scalability Risks 11](#_Toc221522615)

[1.10.6 Time and Schedule Risks 11](#_Toc221522616)

[1.10.7 Requirement Change Risks 11](#_Toc221522617)

[1.10.8 Technical and Development Challenges 11](#_Toc221522618)

[1.11 Summary 11](#_Toc221522619)

[CHAPTER 2 12](#_Toc221522620)

[2.0 LITERATURE REVIEW 12](#_Toc221522621)

[2.1 Evolution of Natural Language Database Interfaces 12](#_Toc221522622)

[2.2 Study of Related Systems 12](#_Toc221522623)

[2.2.1 SQLNet 13](#_Toc221522624)

[2.2.2 Seq2SQL 13](#_Toc221522625)

[2.2.3 Spider Dataset-Based Systems 14](#_Toc221522626)

[2.2.4 OpenAI GPT-Based SQL Assistants 15](#_Toc221522627)

[2.2.5 Supabase AI & Database Assistants 16](#_Toc221522628)

[2.3 Area of Study 17](#_Toc221522629)

[2.3.1 Natural Language Processing (NLP) 17](#_Toc221522630)

[2.3.2 Artificial Intelligence and Machine Learning 17](#_Toc221522631)

[2.3.3 Database Systems 17](#_Toc221522632)

[2.3.4 Human-Computer Interaction (HCI) 17](#_Toc221522633)

[2.3.5 Secure Systems and Access Control 17](#_Toc221522634)

[2.4 Reason for Development 17](#_Toc221522635)

[2.5 Summary 18](#_Toc221522636)

[CHAPTER 3 19](#_Toc221522637)

[3.0 System Requirements 19](#_Toc221522638)

[3.1 Functional Requirements 19](#_Toc221522639)

[3.2 Non-Functional Requirements 20](#_Toc221522640)

[3.2.1 Security 20](#_Toc221522641)

[3.2.2 Performance 20](#_Toc221522642)

[3.2.3 Availability 20](#_Toc221522643)

[3.2.4 Data Privacy 20](#_Toc221522644)

[3.3 Use Case Diagram 20](#_Toc221522645)

[3.3.1 Use Case for User 21](#_Toc221522646)

[3.4 Use Case Description 21](#_Toc221522647)

[3.4.1 Use Case for Query Processing 22](#_Toc221522648)

[3.4.2 Use Case for SQL Generation 23](#_Toc221522649)

[3.4.3 Use Case for Query Execution 24](#_Toc221522650)

[3.4.4 Use Case for Authentication 25](#_Toc221522651)

[3.4.5 Use Case for Admin Monitoring 26](#_Toc221522652)

[3.5 Summary 27](#_Toc221522653)

[CHAPTER 4 28](#_Toc221522654)

[4.0 Methodology 28](#_Toc221522655)

[4.1 Methodology for Software Development 28](#_Toc221522656)

[4.2 Selected Methodology 28](#_Toc221522657)

[4.3 Phases of the Iterative Waterfall Model 29](#_Toc221522658)

[4.3.1 Feasibility Study 29](#_Toc221522659)

[4.3.2 Requirement Analysis 29](#_Toc221522660)

[4.3.3 System Design 30](#_Toc221522661)

[4.3.4 Implementation 30](#_Toc221522662)

[4.3.5 Testing 31](#_Toc221522663)

[4.3.6 Deployment 31](#_Toc221522664)

[4.3.7 Maintenance 31](#_Toc221522665)

[4.4 Reason for Selecting the Iterative Waterfall Model 31](#_Toc221522666)

[4.5 Brief Overview of Talk2SQL System Workflow 31](#_Toc221522667)

[4.5.1 System Design Overview 32](#_Toc221522668)

[4.5.2 Development and Implementation Strategy 32](#_Toc221522669)

[4.6 Summary 32](#_Toc221522670)

[CHAPTER 5 33](#_Toc221522671)

[5.0 Architecture Design 33](#_Toc221522672)

[5.1 Activity Diagram 33](#_Toc221522673)

[5.1.2 Sequence Diagram 35](#_Toc221522674)

[5.1.3 Database Design 36](#_Toc221522675)

[5.1.4 Database Tables Overview 37](#_Toc221522676)

[5.1.5 Design Considerations 37](#_Toc221522677)

[5.1.6 System Architecture 38](#_Toc221522678)

[5.1.7 Architectural Layers 38](#_Toc221522679)

[5.2 Summary 39](#_Toc221522680)

**List of Tables**

**Description Page#**

# LIST OF FIGURES

**Description Page#**

# List of Abbreviations

|  |  |
| --- | --- |
| **Abbreviation** | **Full Form / Contextual Meaning** |
| **AI** | Artificial Intelligence |
| **BI** | Business Intelligence |
| **DBA** | Database Administrator |
| **DCS** | Department of Computer Science |
| **FR** | Functional Requirement |
| **FYP** | Final Year Project |
| **HCI** | Human-Computer Interaction |
| **LLM** | Large Language Model |
| **NL** | Natural Language |
| **NLIDB** | Natural Language Interfaces to Databases |
| **NLP** | Natural Language Processing |
| **NTU** | National Textile University |
| **PII** | Personally Identifiable Information |
| **RAG** | Retrieval-Augmented Generation |
| **SaaS** | Software as a Service |
| **SDG** | Sustainable Development Goal |
| **SQL** | Structured Query Language |
| **UC** | Use Case |
| **UI** | User Interface |

**CHAPTER 1**

## Introduction

In recent years, organizations across industries have become increasingly dependent on data-driven decision making. Relational databases are widely used to store critical business information such as sales records, customer data, inventory, and operational metrics. While the volume of stored data continues to grow rapidly, the ability to access and analyze this data efficiently remains limited to a small group of technically skilled professionals. Most stakeholders, including managers, executives, and domain experts, are unable to directly interact with databases due to the technical complexity of query languages such as SQL. This growing gap between data availability and data accessibility has become a significant challenge for modern organizations.

Traditionally, retrieving meaningful information from a database involves multiple roles and stages. A database administrator is responsible for writing and executing queries, after which a business or data analyst interprets the retrieved data to generate reports, insights, or predictions. This dependency chain not only consumes valuable time but also increases operational costs and delays decision making. Although large language models have demonstrated strong capabilities in data analysis, organizations are often reluctant to share confidential and sensitive business data with cloud-based AI platforms due to privacy, security, and compliance concerns. As a result, many organizations remain constrained by manual workflows despite having access to advanced analytical tools.

Existing solutions such as business intelligence platforms and SQL-based query interfaces attempt to simplify database interaction but still require technical expertise or extensive training. Similarly, approaches that combine databases with general-purpose language models frequently suffer from limitations such as schema hallucination, lack of access control, and potential misuse of organizational data for model training. These limitations make such solutions unsuitable for environments where data privacy, role separation, and system reliability are critical. Consequently, there is a clear need for a secure, accessible, and database-aware system that enables users to retrieve insights without exposing sensitive data or compromising database integrity.

Talk2SQL is designed to address these challenges by enabling users to interact with relational databases using natural language through text or voice-based prompts. The system allows users with basic proficiency in English or Roman Urdu to retrieve data, generate reports, and perform basic analysis without writing SQL queries. Talk2SQL can be deployed locally or on a self-hosted cloud server, ensuring that organizational data remains private and under full administrative control. By utilizing database schema awareness, the system generates accurate and context-aware SQL queries while avoiding the risks associated with generic language models that rely on user data for training.

The platform is implemented as a web-based application with a user-friendly interface and supports widely used database systems such as SQL Server, PostgreSQL, and Supabase. To ensure secure operation, Talk2SQL enforces role-based access control, allowing read-only operations for standard users while reserving write permissions for administrators. Additionally, the system actively prevents destructive database operations and mitigates security vulnerabilities such as SQL injection by validating queries and requiring explicit confirmation for sensitive actions. By simplifying database interaction while maintaining enterprise-grade security and privacy, Talk2SQL contributes toward democratizing data access and enabling safer adoption of conversational AI in real-world organizational environments.

## 1.1 Problem Background

Organizations rely heavily on relational databases to store critical business data, yet retrieving meaningful insights remains a costly and time-consuming process due to dependency on database administrators and data analysts for query execution and interpretation. This multi-step workflow delays decision-making and increases operational expenses, particularly for non-technical stakeholders who cannot directly interact with SQL-based systems. While AI-driven analytics and large language models offer potential automation, their adoption is limited by data privacy, confidentiality, and compliance concerns, as organizational data often needs to be shared with external cloud platforms. Existing tools also assume technical proficiency, frequently exposing SQL queries or complex analytical workflows that add cognitive and training overhead. Furthermore, the lack of strict governance mechanisms such as enforced role-based access control, read-only execution, and protection against destructive queries introduces security risks, including accidental data modification and SQL injection vulnerabilities. Consequently, organizations face a persistent gap between data availability and data usability, highlighting the need for a secure, efficient, and privacy-preserving system that enables natural language interaction with relational databases without compromising data integrity.

## 1.3 Purpose

The purpose of this project is to develop a secure, privacy-preserving system that allows users to interact with relational databases using natural language through text or voice prompts. Talk2SQL aims to empower non-technical stakeholders to retrieve insights, generate basic reports, and perform fundamental data analysis without relying on database administrators or analysts. By eliminating the dependency on SQL expertise, the system accelerates decision-making, reduces operational costs, and ensures that organizational data remains confidential, compliant, and protected against unauthorized or destructive operations. This project demonstrates how conversational AI can be safely integrated with enterprise databases to democratize access to information while maintaining full governance and security controls.

## 1.4 Scope

The scope of this project encompasses the development of a web-based platform that interfaces with relational databases including SQL Server, PostgreSQL, and Supabase. It supports natural language querying in English and Roman Urdu, along with voice input for greater accessibility. The system implements role-based access control, allowing read-only operations for general users and controlled write access for administrators. Talk2SQL generates schema-aware SQL queries to provide accurate insights and prevents destructive operations through query validation and confirmation prompts. Security measures, including safeguards against SQL injection, are integrated to maintain database integrity. The project focuses on structured data and basic analytical tasks, and does not include unstructured data analysis, advanced predictive modeling, or integration with external cloud-based AI services.

## 1.5 Objectives

1. To develop an AI-powered web-based system that enables natural language querying of relational SQL databases.
2. To design and implement multilingual query processing capabilities, supporting both English and Roman Urdu languages for user interaction.
3. To enable non-technical users to generate downloadable reports and perform basic data analysis through interactive visualizations, supporting intelligent, data-driven decision-making.
4. To integrate Retrieval-Augmented Generation (RAG) and contextual AI models to enhance query accuracy, minimize hallucinations, and ensure the generation of precise, schema-aware SQL queries aligned with real-time organizational data.

## 1.6 Research Goals

The Talk2SQL system aims to simplify and secure access to organizational data by enabling natural language interaction with relational databases. The project focuses on reducing technical dependency, minimizing operational costs, and ensuring privacy-preserving data access while maintaining enterprise-grade security and governance. Talk2SQL establishes the following research goals:

### 1.6.1 Improving Data Accessibility and Decision-Making

* Enable non-technical users to retrieve insights from relational databases using natural language queries through text and voice interfaces.
* Reduce dependency on database administrators and data analysts, thereby minimizing delays in decision-making and improving organizational efficiency.
* Support multilingual interaction, including English and Roman Urdu, to enhance accessibility for a broader user base.

### 1.6.2 Enhancing Data Privacy and Security

* Ensure that organizational data, including sensitive business information and personally identifiable information (PII), remain within local or self-hosted environments.
* Prevent unauthorized data access by enforcing strict role-based access control, allowing read-only operations for standard users and controlled write operations for administrators.
* Mitigate security risks such as SQL injection and accidental data modification through query validation and confirmation mechanisms [9].

### 1.6.3 Advancing Intelligent and Responsible AI Integration

* Utilize schema-aware query generation to improve the accuracy and reliability of natural language to SQL translation.
* Avoid the use of organizational data for model training, addressing privacy and compliance concerns associated with cloud-based large language models.
* Demonstrate the feasibility of integrating conversational AI with production databases in a controlled and responsible manner.

### 1.6.4 Reducing Operational Cost and Time Overhead

* Minimize the cost associated with manual data retrieval and analysis by automating query generation and result interpretation.
* Decrease time spent on routine data requests, enabling technical teams to focus on higher-value tasks.
* Provide faster access to insights, supporting timely business decisions and improving overall productivity.

### 1.6.5 Contributing to Global Digital and Sustainable Development Goals

Talk2SQL aligns with global initiatives focused on responsible technology adoption, digital inclusion, and sustainable organizational growth.



Figure 1. 1 The goals for sustainable development

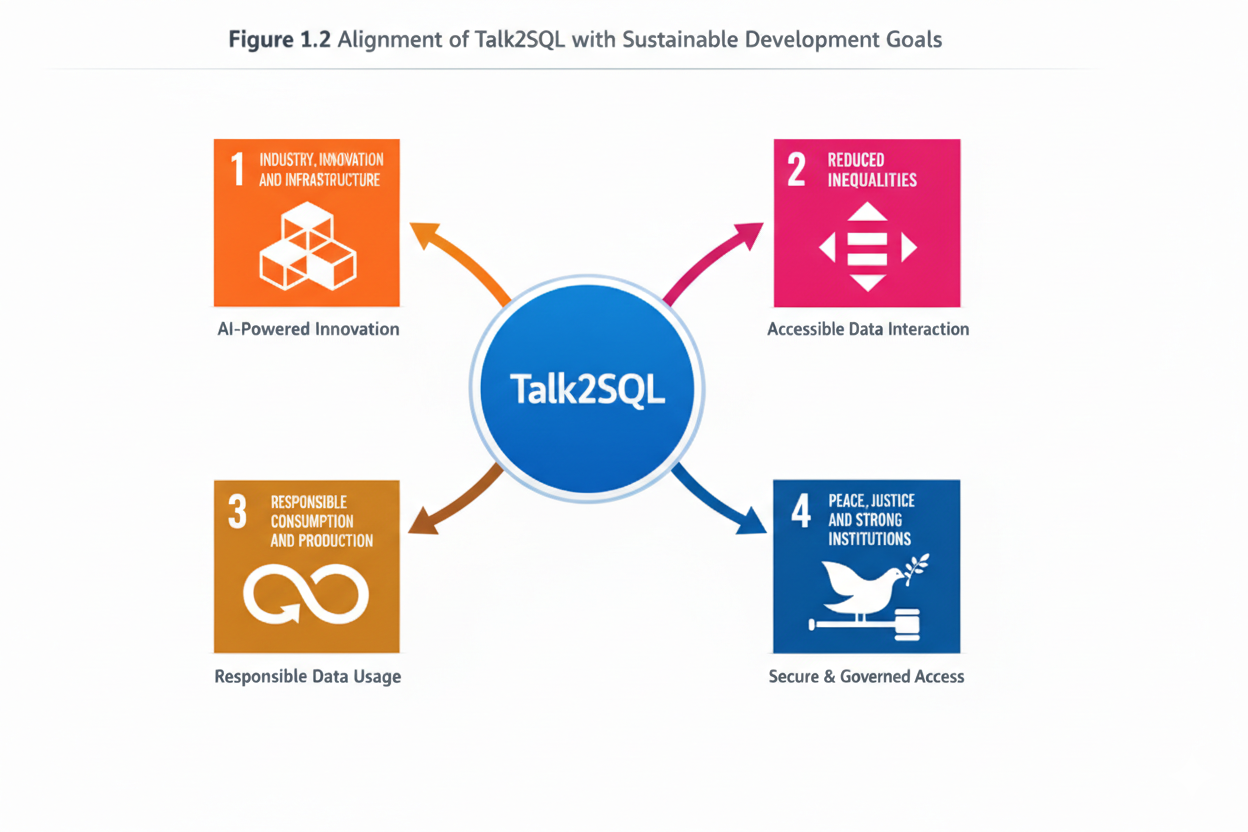


Figure 1. 2 Alignment of Talk2SQL with sustainable development goals

#### 1.6.5.1 Align with SDG 9 – Industry, Innovation, and Infrastructure

* Promote innovation by enabling AI-powered, secure, and scalable data access systems within organizational infrastructures.

#### 1.6.5.2 Support SDG 10 – Reduced Inequalities

* Reduce technological barriers by providing accessible data interaction tools for non-technical and linguistically diverse users.

#### 1.6.5.3 Contribute to SDG 12 – Responsible Consumption and Production

* Encourage responsible use of data and AI systems by prioritizing privacy, governance, and controlled access to information resources.

#### 1.6.5.4 Support SDG 16 – Peace, Justice, and Strong Institutions

* Strengthen organizational transparency and accountability by enabling secure and governed access to data for informed decision-making.

## 1.7 Cost Benefit Analysis of Talk2SQL

The cost analysis of Talk2SQL evaluates its financial feasibility in comparison with popular natural language database query tools and traditional data analytics systems. It highlights affordability, scalability, and potential cost savings for organizations aiming to democratize data access while maintaining security and governance.

### 1.7.1 Cost Comparison with Existing Database AI and Analytics Tools

This section compares the estimated costs of Talk2SQL with existing database AI query systems, detailing their pricing models and associated costs.

#### 1.7.1.1 AskYourDatabase

* Offers cloud-based and desktop NLP SQL chatbot interfaces.
* Desktop plans start around **$49 per month**, while scalable website chatbot plans range from **$149 to $329 per month**, with enterprise pricing available by custom quote.

#### 1.7.1.2 Querio

* Enterprise-focused tool with governance and BI integrations.
* Core platform pricing starts at approximately **$14,000 per year**, with add-ons (dashboards, extra database connections) costing additional thousands annually. Self-hosted enterprise deployments can exceed **$60,000 per year**.

#### 1.7.1.3 DB Pilot

* Desktop AI SQL assistant and editor.
* Subscription plans typically cost around **$20 per month**, and a one-time purchase option is available at **$79**.

#### 1.7.1.4 AI2SQL / Text2SQL.ai

* Web-based SQL generation tools with varied pricing.
* Entry-level plans often fall between **$9 and $39 per month** depending on query volume and database connectors.

#### 1.7.2 Talk2SQL: A Cost-Effective Alternative

Unlike many existing tools that emphasize SQL composition, dashboards, or large enterprise fees, Talk2SQL is positioned to reduce total cost of ownership by focusing on secure natural language database access with strong governance.

#### 1.7.2.1 No Per-User Licensing Overhead

* Instead of charging per seat or per query limits like many SaaS tools, Talk2SQL’s value comes from a **self-hosted or enterprise deployment** model that limits ongoing subscription costs, particularly for organizations with many users.

#### 1.7.2.2 Avoids Cloud AI Data Costs

* Tools that rely on cloud APIs or hosted AI often incur usage fees for token processing and storage, in addition to subscription costs expenses that can escalate rapidly with query volume. Talk2SQL’s on-prem/self-hosted design helps avoid ongoing cloud compute charges.

#### 1.7.2.3 Reduces Operational Dependencies

* Traditional reliance on database administrators and analysts can result in personnel costs (e.g., an experienced DBA can command six-figure salaries). Automating query access reduces repetitive task costs and frees technical staff for higher-value work.

#### 1.7.2.4 Minimal Maintenance and Infrastructure Costs

* Without expensive hardware requirements, robotics components, or dedicated BI infrastructure, Talk2SQL mainly incurs **software development and occasional update costs** rather than ongoing hardware maintenance.

### 1.7.3 Affordability and Scalability

This section explores how Talk2SQL remains cost-effective across organizational sizes and deployment scenarios.

#### 1.7.3.1 Suitable for Small and Medium Enterprises

* With many BI tools costing **tens of thousands per year for modest user groups**, smaller businesses can adopt Talk2SQL at a fraction of that cost while gaining secure natural language access to legacy databases.

#### 1.7.3.2 Low Total Cost of Ownership

* By reducing dependency on high-cost licenses, external data transfers, and dedicated specialists for everyday querying tasks, Talk2SQL lowers the total cost over time compared with traditional BI platforms and enterprise tools whose annual support fees can exceed hundreds of thousands.

#### 1.7.3.3 Scalable from Desktop to Enterprise Deployments

* Organizations can begin with local or departmental installations and scale to larger deployments without proportional increases in subscription expenditures, ensuring predictable budgeting.

## ****1.8 Proposed Solution****

Talk2SQL is an AI-powered natural language interface that enables users to interact with relational databases using plain English instead of writing complex SQL queries. The system bridges the gap between non-technical users and structured databases by converting natural language questions into secure, optimized SQL queries and returning meaningful results in real time.

The proposed solution consists of an AI-driven query engine, a secure backend connected to databases, and a user-friendly web interface for seamless interaction.

### 1.8.1 AI-Powered Natural Language Interface for Users

The core component of Talk2SQL is an AI-powered interface that allows users to query databases without SQL knowledge. The system focuses on simplicity, accuracy, and security.

#### 1.8.1.1 Natural Language Querying

* Users can ask questions such as *“Show total sales for last month”* or *“List employees hired after 2022”*.
* The system interprets user intent and automatically generates the corresponding SQL query.

#### 1.8.1.2 SQL Query Generation

* Advanced NLP and LLM-based models are used to convert natural language input into syntactically correct and optimized SQL queries.
* The generated queries are validated before execution to prevent errors and malicious operations.

#### 1.8.1.3 Context Awareness

* The system maintains context across queries, allowing follow-up questions like *“Now group it by department”* without repeating the entire query.
* Schema awareness ensures correct table and column mapping.

### 1.8.2 Secure Backend System

The backend acts as a controlled layer between the user interface and the database to ensure safety and reliability.

#### 1.8.2.1 Query Validation and Access Control

* Read-only or role-based access policies ensure users can only query permitted tables and columns.
* Sensitive operations such as DELETE, UPDATE, or DROP are restricted unless explicitly authorized.

#### 1.8.2.2 Database Compatibility

* Talk2SQL supports popular relational databases such as PostgreSQL, MySQL, and Supabase-based databases.
* Schema introspection is used to dynamically adapt to different database structures.

### 1.8.3 Result Interpretation and Visualization

Instead of returning raw query outputs, Talk2SQL presents results in a human-friendly format.

* Query results are displayed as tables, summaries, or aggregated values.
* Numerical results can be accompanied by short AI-generated explanations to improve understanding.
* This reduces cognitive load for non-technical users and supports faster decision-making.

### 1.8.4 Role-Based User Management

The system supports multiple user roles to ensure proper governance.

* **Admin Users:** Full database access and permission control.
* **Business Users:** Limited access for analytics and reporting.
* **Read-Only Users:** Can view insights without modifying data.

This structure ensures secure collaboration across teams without risking data integrity.

### 1.8.5 Development of AI Models

The AI component of Talk2SQL is built around customized NLP and language models optimized for database interaction.

* Models are trained to understand schema relationships, joins, filters, and aggregations.
* Error-handling logic reduces incorrect query generation and improves reliability.
* Continuous refinement improves accuracy for domain-specific databases.

### 1.8.6 User Interface Development

The Talk2SQL web interface is designed with usability and clarity as top priorities.

* Clean, minimal layout with a conversational query input box.
* Clear display of results, generated SQL (optional), and execution status.
* Designed for both technical and non-technical users, enabling fast onboarding with minimal learning curve.

The interface ensures that users can confidently explore and analyze data without requiring database expertise.

## 1.9 Project Scheduling

The project scheduling outlines the complete development timeline of **Talk2SQL**, covering all major phases from problem identification to final deployment and documentation. The semester officially commenced on **Monday, 2 February 2026**, and spans a total duration of **17 weeks**, concluding on **Sunday, 31 May 2026**.

A structured schedule is used to clearly define the **start date, end date, and duration** of each project activity. This ensures systematic progress, effective resource utilization, and timely achievement of milestones. The detailed project schedule is presented in **Figure 1.2 Project Timeline**.

Timeline of Talk2SQL

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Task No.** | **Activity** | **Start Date** | **End Date** | **Duration** |
| 1 | Problem Identification & Requirement Analysis | 02 Feb 2026 | 08 Feb 2026 | 1 Week |
| 2 | Literature Review & Feasibility Study | 09 Feb 2026 | 22 Feb 2026 | 2 Weeks |
| 3 | System Design & Architecture Planning | 23 Feb 2026 | 08 Mar 2026 | 2 Weeks |
| 4 | Dataset & Database Schema Preparation | 09 Mar 2026 | 22 Mar 2026 | 2 Weeks |
| 5 | AI Model Selection & NLP Pipeline Design | 23 Mar 2026 | 05 Apr 2026 | 2 Weeks |
| 6 | Talk2SQL Core Development (NL → SQL Engine) | 06 Apr 2026 | 26 Apr 2026 | 3 Weeks |
| 7 | Backend Integration & Security Implementation | 27 Apr 2026 | 10 May 2026 | 2 Weeks |
| 8 | Frontend Development & UI Integration | 11 May 2026 | 17 May 2026 | 1 Week |
| 9 | Testing, Optimization & Error Handling | 18 May 2026 | 24 May 2026 | 1 Week |
| 10 | Documentation, Final Review & Submission | 25 May 2026 | 31 May 2026 | 1 Week |

## 1.10 Risk Management

The development of the **Talk2SQL** project involves several technical, operational, and organizational risks that may affect its successful implementation. Identifying these risks early allows effective planning and mitigation to ensure system reliability, security, and performance.

### 1.10.1 Algorithm and AI Model Risk

The core functionality of Talk2SQL depends on the accuracy of Natural Language Processing (NLP) and SQL generation models. If the AI model is trained on insufficient, biased, or low-quality datasets, it may generate incorrect or inefficient SQL queries. Such inaccuracies can lead to wrong data retrieval, misinterpretation of user intent, and reduced system reliability.

### 1.10.2 Security and Privacy Risks

Talk2SQL interacts with sensitive organizational databases that may contain confidential business or personal data. Unauthorized access, SQL injection attacks, or improper permission handling could result in data breaches. Ensuring secure authentication, role-based access control, and encrypted communication is critical to protecting data privacy.

### 1.10.3 User Interface and Experience Risk

Since Talk2SQL is designed for non-technical users, the usability of the interface plays a vital role. A complex or unintuitive UI may discourage adoption and increase user errors while formulating queries. Poor user experience can limit the system’s effectiveness despite strong backend performance.

### 1.10.4 Integration and Compatibility Risks

The system must integrate seamlessly with different database management systems and backend services. Compatibility issues with database schemas, SQL dialects, or APIs may lead to incorrect query execution or system instability, affecting overall functionality.

### 1.10.5 Performance and Scalability Risks

Talk2SQL may be required to handle multiple concurrent users and complex queries in real time. Performance degradation, slow query generation, or system crashes under high load could reduce reliability. Additionally, the system must be scalable to support growing datasets and future feature expansions.

### 1.10.6 Time and Schedule Risks

Delays may occur due to unforeseen technical challenges, model fine-tuning, or integration issues. Such delays can impact milestone completion and overall project delivery. Proper scheduling, task tracking, and buffer planning are essential to minimize this risk.

### 1.10.7 Requirement Change Risks

User expectations and project requirements may evolve during development, leading to scope changes. Frequent modifications can cause scope creep, increased development time, and resource strain if not managed properly.

### 1.10.8 Technical and Development Challenges

Developing accurate NLP models, mapping natural language to optimized SQL queries, and ensuring database correctness require advanced technical expertise. Issues such as model hallucination, ambiguous query interpretation, or database schema mismatch may affect system accuracy. Continuous testing and validation are necessary to mitigate these challenges.

## 1.11 Summary

This chapter presents a comprehensive overview of the **Talk2SQL** project, outlining its motivation, objectives, and problem domain. It highlights the challenges associated with traditional database querying, including technical complexity, time-consuming analysis, and privacy concerns. The proposed solution emphasizes natural language interaction, secure database access, and efficient query generation to bridge the gap between non-technical users and structured data systems. This chapter establishes a strong foundation for subsequent discussions on system design, implementation, and evaluation.

# CHAPTER 2

## 2.0 LITERATURE REVIEW

The rapid growth of data-driven systems has made databases a central component of almost every modern application. While databases store structured information efficiently, interacting with them traditionally requires expertise in Structured Query Language (SQL). This technical barrier limits direct database access to developers and database administrators, excluding non-technical users such as managers, analysts, and decision-makers.

To address this challenge, **Natural Language Interfaces to Databases (NLIDB)** emerged as a research area that enables users to query databases using everyday language instead of formal SQL syntax. With advancements in Artificial Intelligence (AI), Natural Language Processing (NLP), and Large Language Models (LLMs), NLIDB systems have evolved significantly, giving rise to modern **Text-to-SQL** or **Talk2SQL** systems.

Talk2SQL builds upon decades of research in database querying, NLP, and machine learning, aiming to bridge the communication gap between humans and structured data systems. This chapter reviews existing research, tools, and systems related to natural language database querying, highlighting their strengths, limitations, and relevance to the proposed Talk2SQL system.

## 2.1 Evolution of Natural Language Database Interfaces

Early attempts at natural language database access date back to the 1970s, where rule-based systems translated predefined linguistic patterns into SQL queries. These systems were limited by rigid grammar rules and required extensive manual configuration. As databases grew in complexity, traditional rule-based approaches struggled to scale.

The introduction of statistical NLP methods and machine learning enabled more flexible interpretation of user queries. Later, deep learning and transformer-based architectures revolutionized this domain by allowing models to learn semantic mappings between natural language questions and SQL queries directly from data.

Modern Talk2SQL systems leverage pre-trained language models, schema understanding, and context-aware parsing to generate accurate SQL queries dynamically. This evolution has shifted NLIDB systems from academic prototypes to real-world, production-ready solutions.

## 2.2 Study of Related Systems

This section reviews existing systems and tools that aim to simplify database querying using natural language or AI-driven approaches.

### 2.2.1 SQLNet

SQLNet is a deep learning-based Text-to-SQL system designed to overcome limitations of sequence-to-sequence models. It eliminates the need to generate SQL queries token by token and instead predicts query components independently.

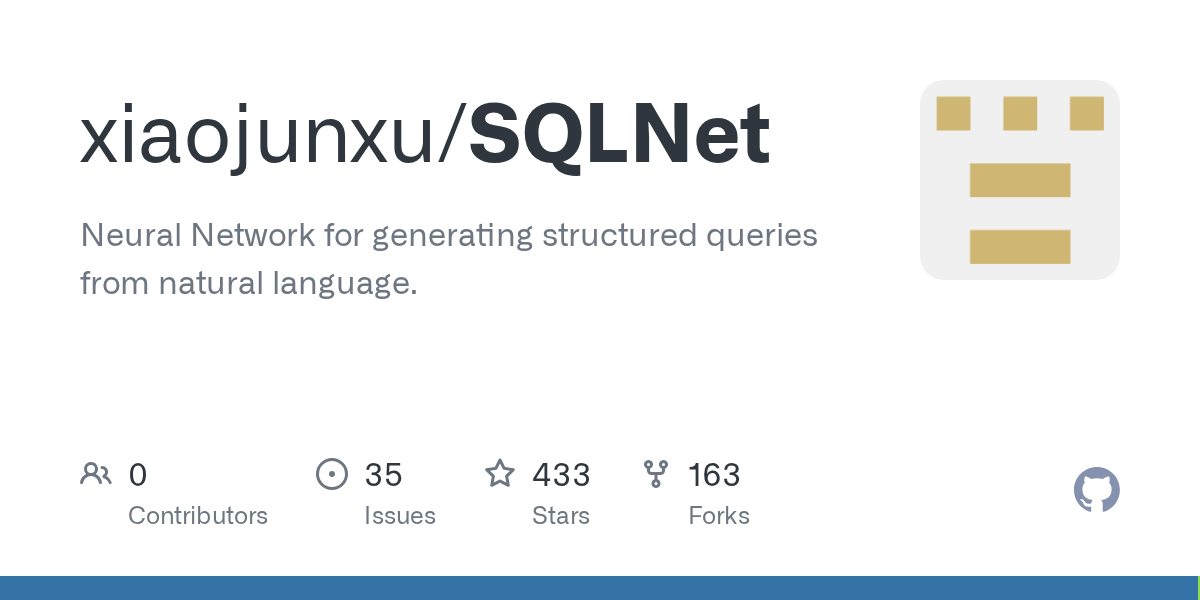


Figure 2. 1 SQLNet

#### 2.2.1.1 Key Features:

* Uses a sketch-based approach
* Reduces syntax errors in generated SQL
* Trained on structured datasets like WikiSQL

#### 2.2.1.2 Limitations:

* Limited support for complex queries
* Struggles with nested queries and joins

SQLNet laid foundational work for learning structured SQL generation without strict sequential dependencies.

### 2.2.2 Seq2SQL

Seq2SQL introduced reinforcement learning to improve SQL query generation accuracy. It combined sequence-to-sequence learning with execution-guided optimization.

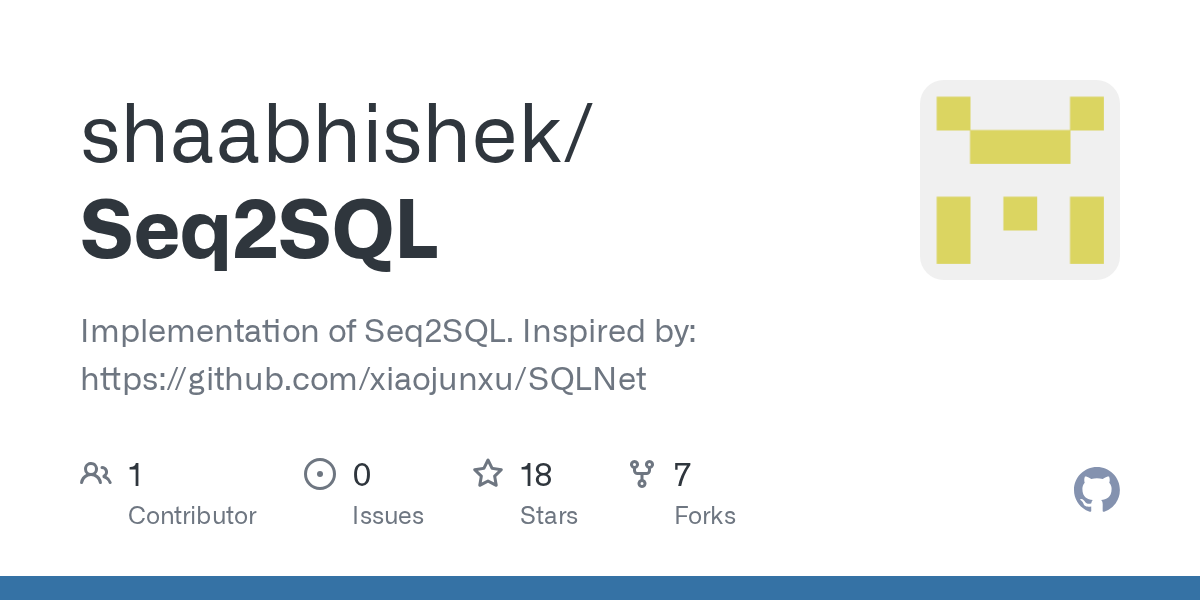


Figure 1. 4 Seq2SQL

#### 2.2.2.1 Key Features:

* Uses reinforcement learning based on query execution results
* Improves accuracy over traditional seq2seq models

#### 2.2.2.2 Limitations:

* Requires execution environment during training
* Limited schema generalization

Seq2SQL demonstrated how feedback from database execution could enhance Text-to-SQL performance.

### 2.2.3 Spider Dataset-Based Systems

The Spider dataset marked a major milestone in Text-to-SQL research by introducing complex, multi-table queries across diverse schemas.

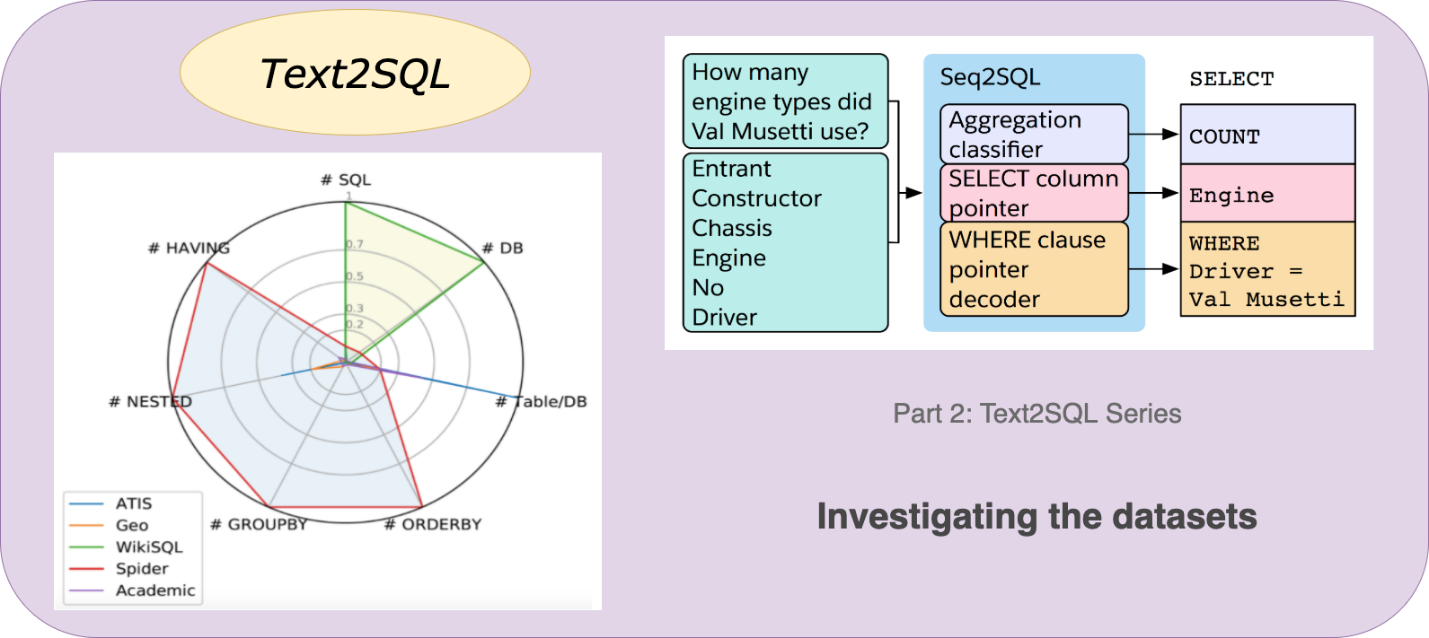


Figure 2.3 TexT2SQL

#### 2.2.3.1 Systems trained on Spider focus on:

* Cross-domain generalization
* Schema linking
* Multi-table joins
* Nested queries

#### 2.2.3.2 Challenges Identified:

* High dependency on schema understanding
* Increased ambiguity in natural language questions

Talk2SQL systems inspired by Spider emphasize robustness across real-world databases.

### 2.2.4 OpenAI GPT-Based SQL Assistants

Large Language Models such as GPT-3, GPT-4, and GPT-4.1 introduced a paradigm shift by enabling zero-shot and few-shot SQL generation.



Figure 2.4 ChatGpt

#### 2.2.4.1 Key Features:

* Understands natural language context deeply
* Generates complex SQL queries without task-specific training
* Handles joins, aggregations, and nested queries

#### 2.2.4.2 Limitations:

* Risk of hallucinated table or column names
* Requires schema injection for accuracy
* Security concerns if directly exposed to production databases

Talk2SQL mitigates these risks through schema validation and controlled query execution.

### 2.2.5 Supabase AI & Database Assistants

Modern platforms like Supabase provide AI-powered query generation tools embedded directly into database dashboards.

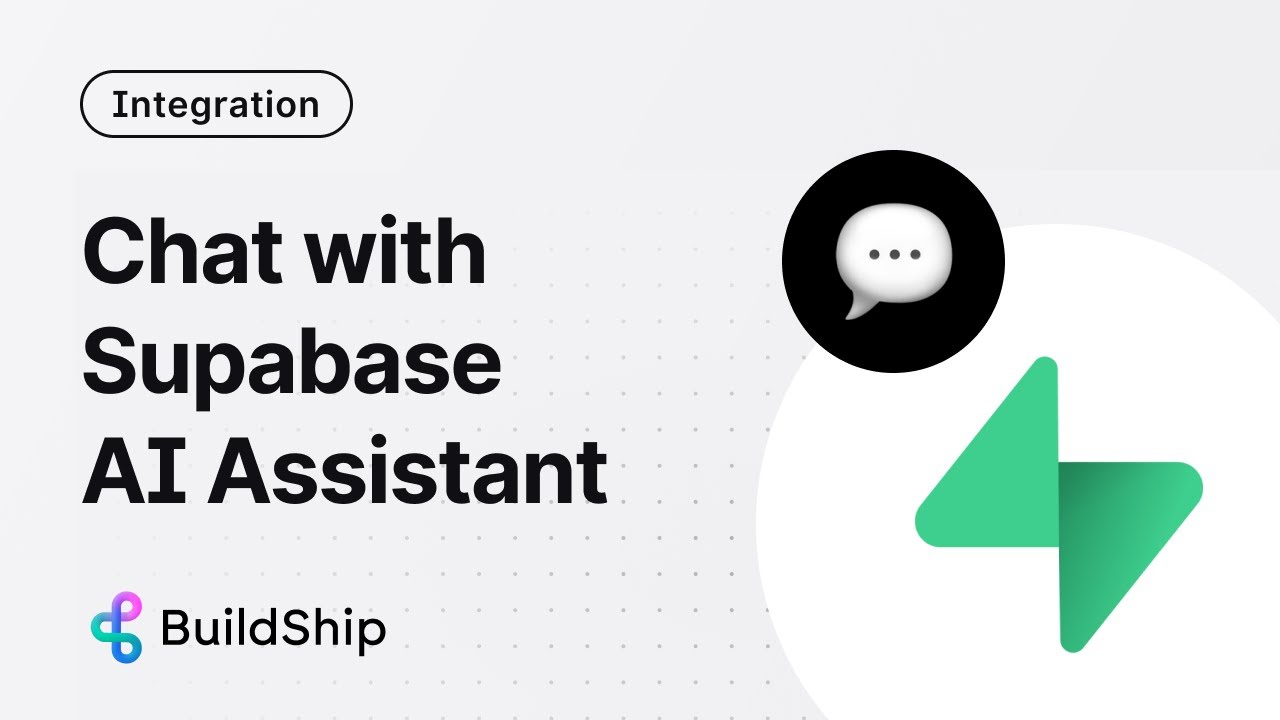


Figure 2.5 Supabase AI Assistant

#### 2.2.5.1 Strengths:

* Schema-aware query generation
* Developer-friendly integration
* Secure execution environment

#### 2.2.5.2 Limitations:

* Limited conversational context
* Focused more on developers than non-technical users

Talk2SQL extends this concept by offering conversational, voice/text-based querying.

## 2.3 Area of Study

The Talk2SQL project intersects multiple research domains:

### 2.3.1 Natural Language Processing (NLP)

#### 2.3.1.1 NLP techniques are used to:

* Tokenize and parse user queries
* Extract intent and entities
* Resolve ambiguity in natural language

### 2.3.2 Artificial Intelligence and Machine Learning

#### 2.3.2.1 Machine learning enables:

* Semantic mapping between text and SQL
* Query optimization suggestions
* Context-aware interpretation of user intent

### 2.3.3 Database Systems

#### 2.3.3.1 Understanding relational database concepts such as:

* Schema design
* Relationships and constraints
* Query execution plans are critical for accurate SQL generation.

### 2.3.4 Human-Computer Interaction (HCI)

2.3.4.1 Talk2SQL emphasizes usability by:

* Reducing technical barriers
* Supporting conversational querying
* Providing understandable responses to users

### 2.3.5 Secure Systems and Access Control

2.3.5.1 The system enforces:

* Read-only or role-based query execution
* Prevention of destructive queries
* Logging and auditability

## 2.4 Reason for Development

Despite significant advancements, existing Text-to-SQL systems face several challenges:

* Lack of conversational continuity
* Poor handling of ambiguous queries
* Limited support for real-world database schemas
* High technical barrier for non-developers

Organizations increasingly rely on data-driven decisions, yet many stakeholders cannot query databases independently. Talk2SQL addresses this gap by providing a **safe, conversational, schema-aware, and user-friendly interface** for database interaction.

By integrating modern LLMs with controlled execution logic and schema grounding, Talk2SQL ensures both **accuracy and security**, making it suitable for enterprise and academic environments.

Table 2.1 Critical Analysis of Existing Systems

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **System** | **Natural Language Support** | **Schema Awareness** | **Security** | **Limitations** |
| SQLNet | Moderate | Limited | Low | No complex queries |
| Seq2SQL | Moderate | Limited | Low | Training complexity |
| GPT-based Tools | High | Medium | Medium | Hallucination risk |
| Supabase AI | High | High | High | Limited conversation |
| **Talk2SQL** | **Very High** | **High** | **High** | Requires schema setup |

Talk2SQL improves upon existing systems by combining conversational intelligence with strict schema validation and controlled SQL execution.

## 2.5 Summary

This chapter reviewed the evolution of natural language database interfaces, highlighting major research contributions and modern AI-driven systems. Existing solutions demonstrate the feasibility of Text-to-SQL systems but expose limitations in usability, security, and real-world adaptability.

Talk2SQL builds upon these foundations by integrating conversational AI, schema-aware SQL generation, and secure execution mechanisms. This literature review establishes the theoretical and practical basis for the proposed system, leading into system requirements and architectural design discussed in subsequent chapters.

# CHAPTER 3

## 3.0 System Requirements

Talk2SQL is an intelligent Natural Language Processing (NLP)–based system designed to bridge the gap between non-technical users and relational databases. The system enables users to interact with structured databases using natural language queries and automatically translates them into optimized SQL queries.

This chapter outlines the complete system requirements of the Talk2SQL application. It defines both **functional** and **non-functional** requirements, ensuring clarity in system behavior, performance expectations, security constraints, and usability standards. Additionally, use case diagrams and detailed use case descriptions are presented to illustrate interactions between users and the system.

## 3.1 Functional Requirements

Functional requirements describe what the Talk2SQL system must do. These requirements define the core features necessary to convert natural language queries into SQL queries accurately and securely while ensuring a smooth user experience.

Table 3.1 User Requirements

|  |  |
| --- | --- |
| **Requirement No** | **Description** |
| FR-1 | Allow users to input queries in natural language |
| FR-2 | Translate natural language queries into valid SQL queries |
| FR-3 | Execute generated SQL queries on the connected database |
| FR-4 | Display query results in a structured tabular format |
| FR-5 | Allow users to view and copy generated SQL queries |

Table 3.2 Admin Requirements

|  |  |
| --- | --- |
| **Requirement No** | **Description** |
| FR-1 | Manage database connections and schemas |
| FR-2 | Monitor user query logs |
| FR-3 | Control access permissions |
| FR-4 | Manage system configurations and models |

Table 3.3 System Requirements

|  |  |
| --- | --- |
| **Requirement No** | **Description** |
| FR-1 | Integrate NLP model for intent and entity extraction |
| FR-2 | Map extracted intents to SQL templates |
| FR-3 | Validate SQL queries before execution |
| FR-4 | Support multiple relational databases |

Table 3.4 Web Application Requirements

|  |  |
| --- | --- |
| **Requirement No** | **Description** |
| FR-1 | User authentication and authorization |
| FR-2 | Interactive query input interface |
| FR-3 | Display SQL query and execution results |
| FR-4 | Error handling and query feedback |

Table 3.5 NLP Engine Requirements

|  |  |
| --- | --- |
| **Requirement No** | **Description** |
| FR-1 | Perform tokenization and entity recognition |
| FR-2 | Identify tables, columns, and conditions |
| FR-3 | Generate syntactically correct SQL |
| FR-4 | Handle ambiguous queries intelligently |

## 3.2 Non-Functional Requirements

### 3.2.1 Security

The Talk2SQL system enforces strict security mechanisms to protect database integrity and user data. Authentication and role-based access control prevent unauthorized query execution. SQL injection prevention, query sanitization, and permission checks ensure secure database interaction.

### 3.2.2 Performance

The system is designed to generate SQL queries and return results with minimal latency. NLP processing and query execution are optimized to ensure real-time responsiveness, even with complex queries and large datasets.

### 3.2.3 Availability

Talk2SQL ensures high availability through reliable backend services and cloud-based deployment. System uptime is maintained through proper error handling and scalable infrastructure.

### 3.2.4 Data Privacy

User queries, database credentials, and results are handled confidentially. Sensitive data is encrypted, and logs are maintained strictly for debugging and analytics while complying with data protection regulations.

## 3.3 Use Case Diagram

Use case diagrams provide a visual representation of interactions between system actors and Talk2SQL functionalities. The primary actors are **User** and **Admin**.

### 3.3.1 Use Case for User

**Figure 3.1 Use Case for User** illustrates how users interact with Talk2SQL to submit natural language queries, view generated SQL, and retrieve query results.

**3.3.2 Use Case for Admin**

**Figure 3.1 Use Case for Admin** demonstrates administrative interactions such as managing schemas, monitoring logs, and controlling system settings.

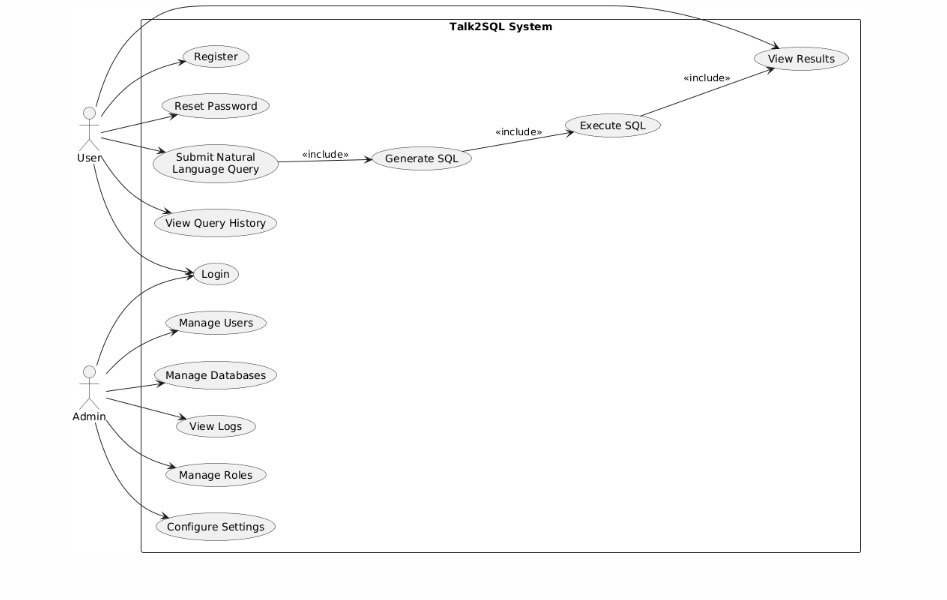


Figure 3.1 Use case for User and Admin

## 3.4 Use Case Description

Each use case is described with preconditions, postconditions, actors, and activity flows.

### 3.4.1 Use Case for Query Processing

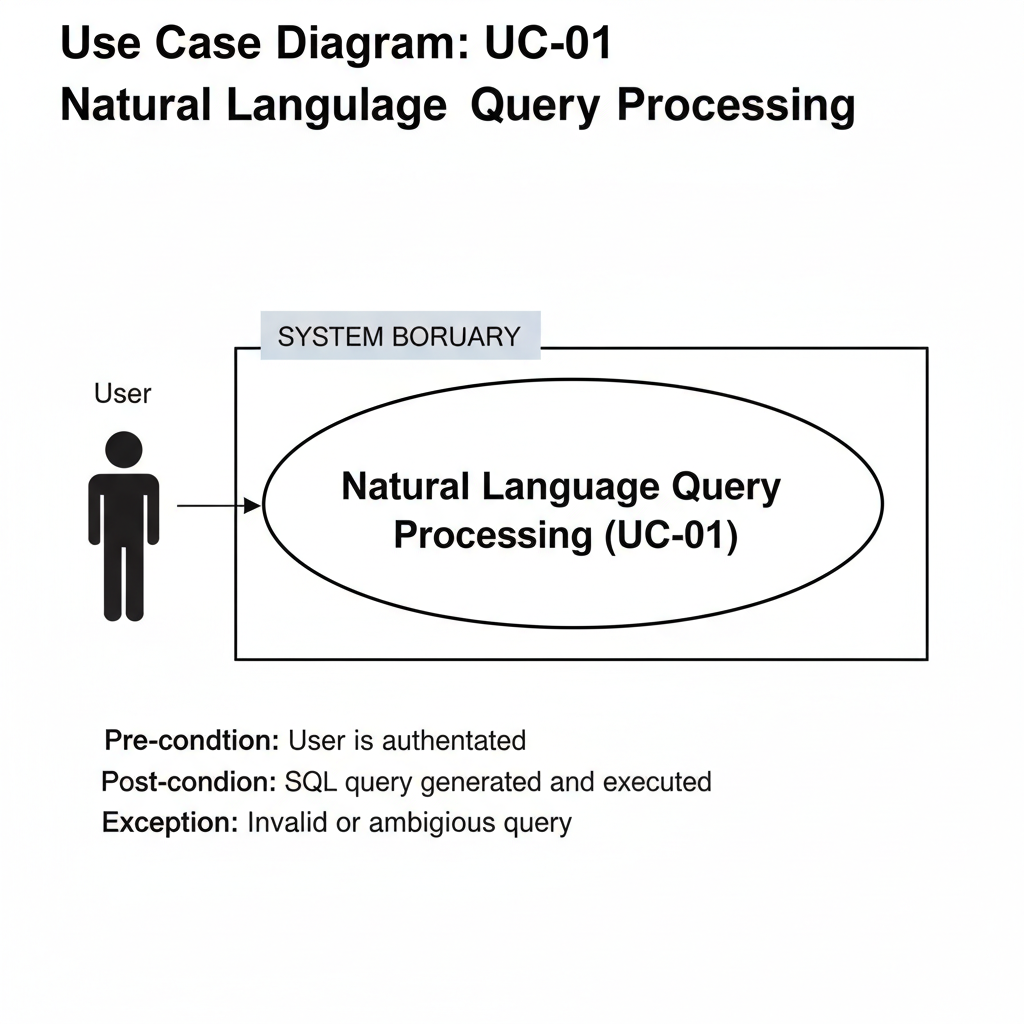


Figure 3.2 Use-Case for Natural Language Query Processing

Table 3.6 Use Case for Query Processing

|  |  |
| --- | --- |
| Field | Description |
| Use Case ID | UC-01 |
| Use Case Name | Natural Language Query Processing |
| Actor | User |
| Description | User submits a natural language query |
| Pre-condition | User is authenticated |
| Post-condition | SQL query generated and executed |
| Exception | Invalid or ambiguous query |

### 3.4.2 Use Case for SQL Generation

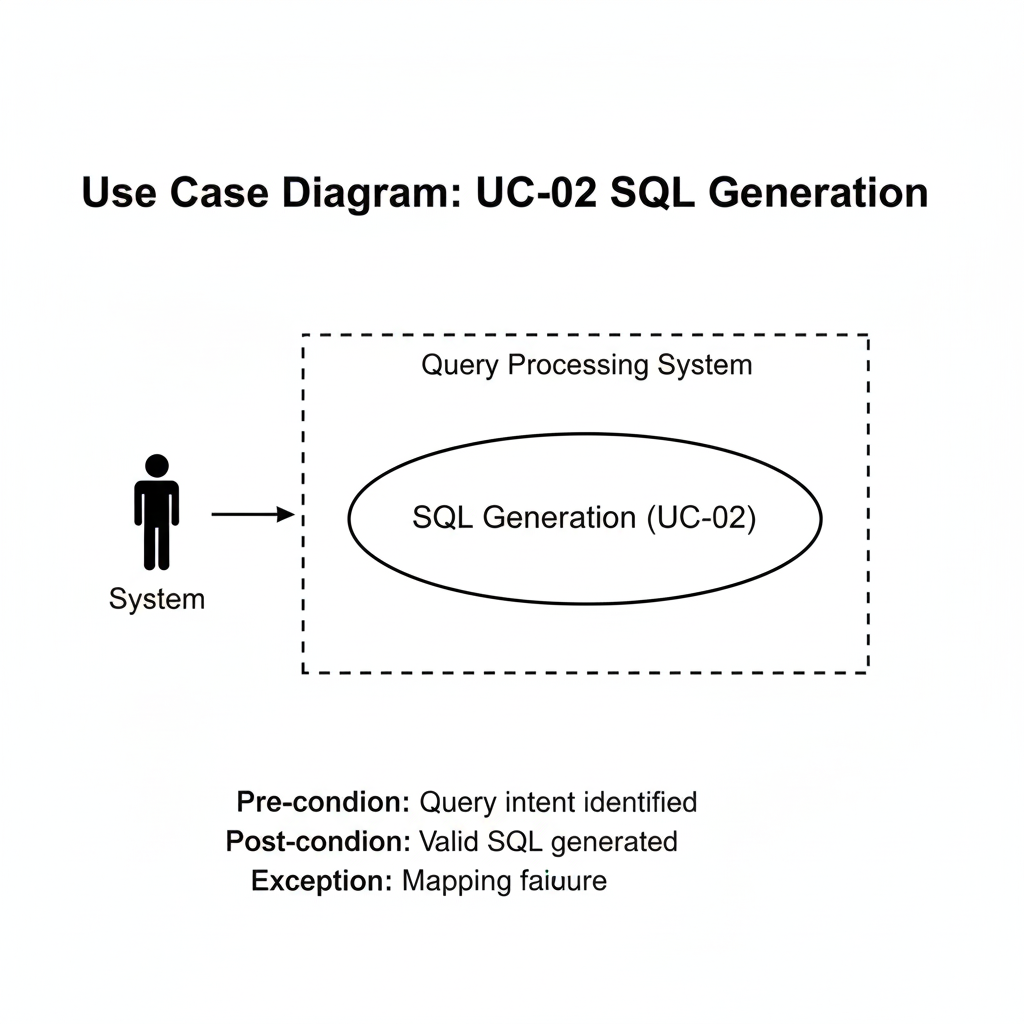


Figure 3.3 Use Case for SQL Query Generation

Table 3.7 Use Case for SQL Generation

|  |  |
| --- | --- |
| **Field** | **Description** |
| Use Case ID | UC-02 |
| Use Case Name | SQL Generation |
| Actor | System |
| Description | Convert NLP output to SQL |
| Pre-condition | Query intent identified |
| Post-condition | Valid SQL generated |
| Exception | Mapping failure |

### 3.4.3 Use Case for Query Execution

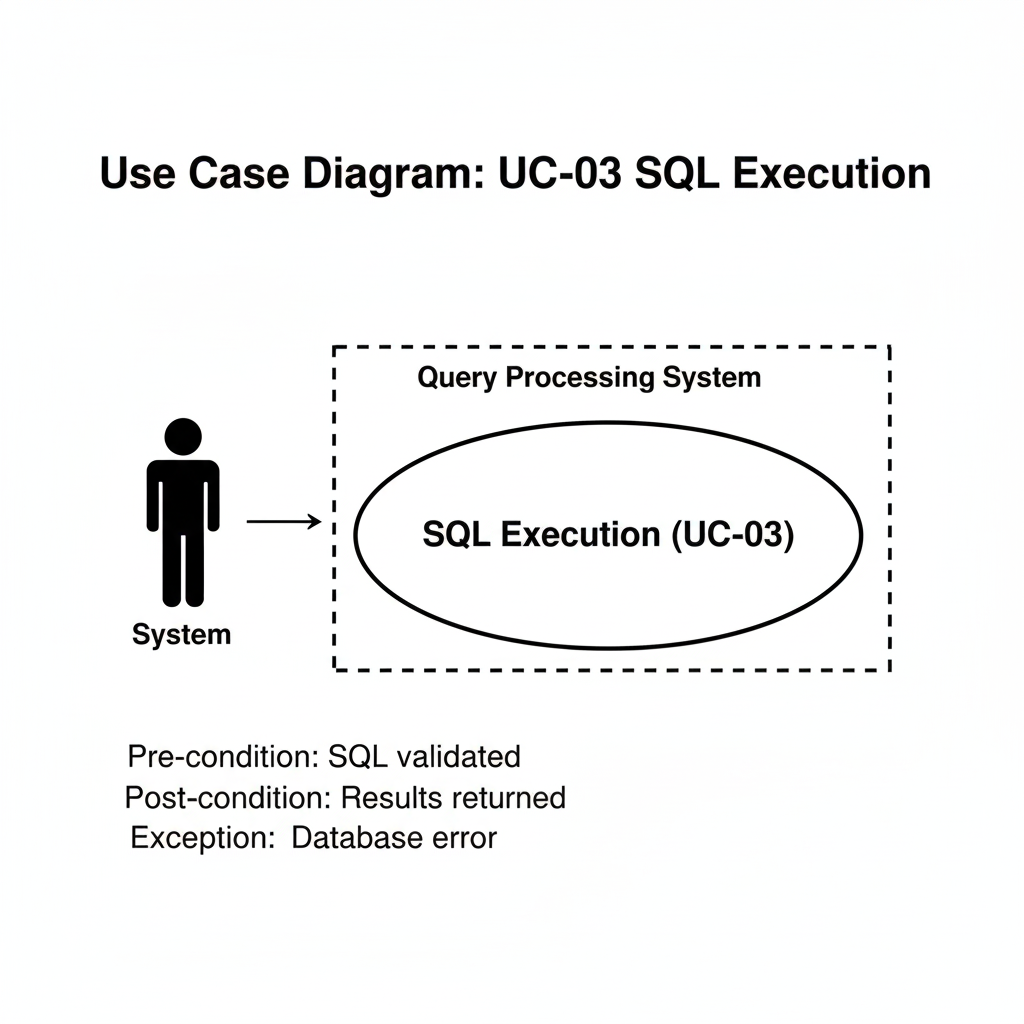


Figure 3.4 Use Case for SQL Execution

Table 3.8 Use Case for Query Execution

|  |  |
| --- | --- |
| **Field** | **Description** |
| Use Case ID | UC-03 |
| Use Case Name | SQL Execution |
| Actor | System |
| Description | Execute SQL on database |
| Pre-condition | SQL validated |
| Post-condition | Results returned |
| Exception | Database error |

### 3.4.4 Use Case for Authentication

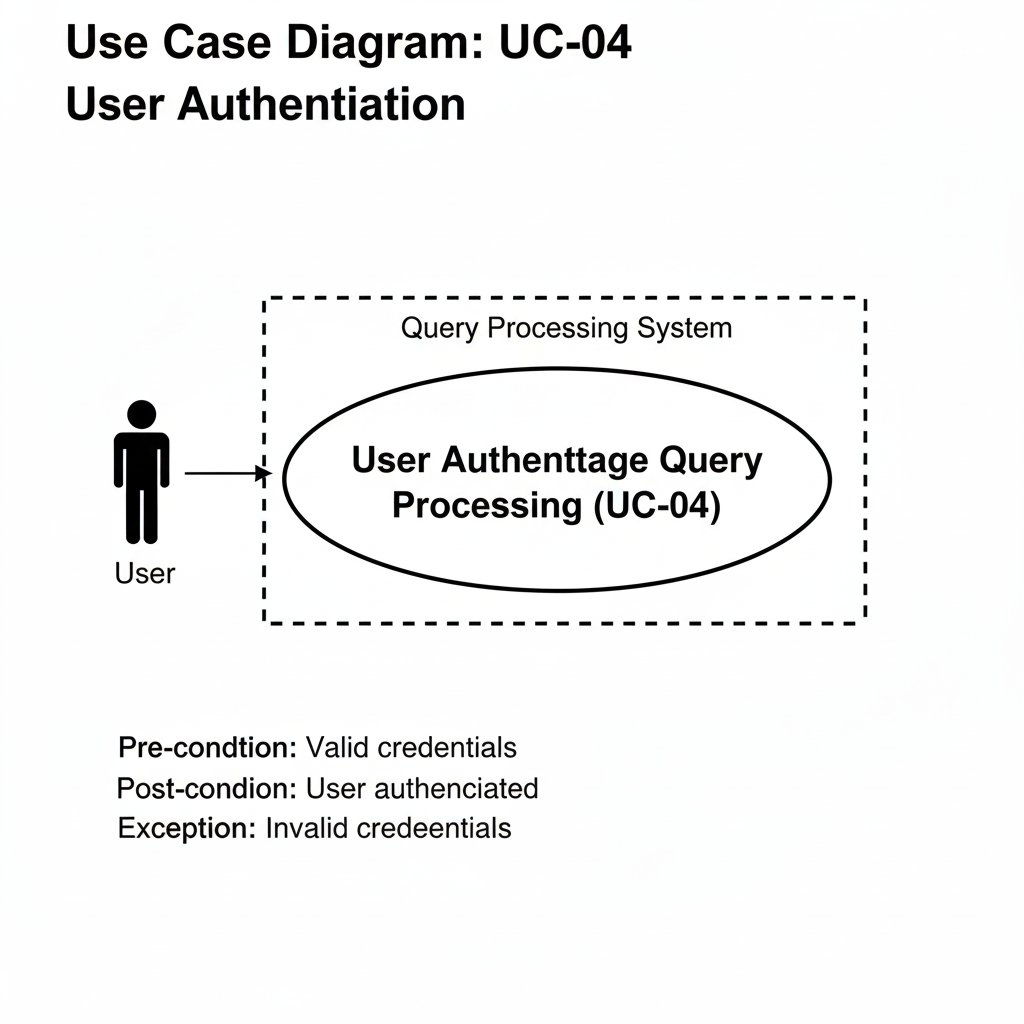


Figure 3.5 Use Case for User Authentication

Table 3.9 Use Case for Authentication

|  |  |
| --- | --- |
| **Field** | **Description** |
| Use Case ID | UC-04 |
| Use Case Name | User Authentication |
| Actor | User |
| Description | Login and session management |
| Pre-condition | Valid credentials |
| Post-condition | User authenticated |
| Exception | Invalid credentials |

### 3.4.5 Use Case for Admin Monitoring

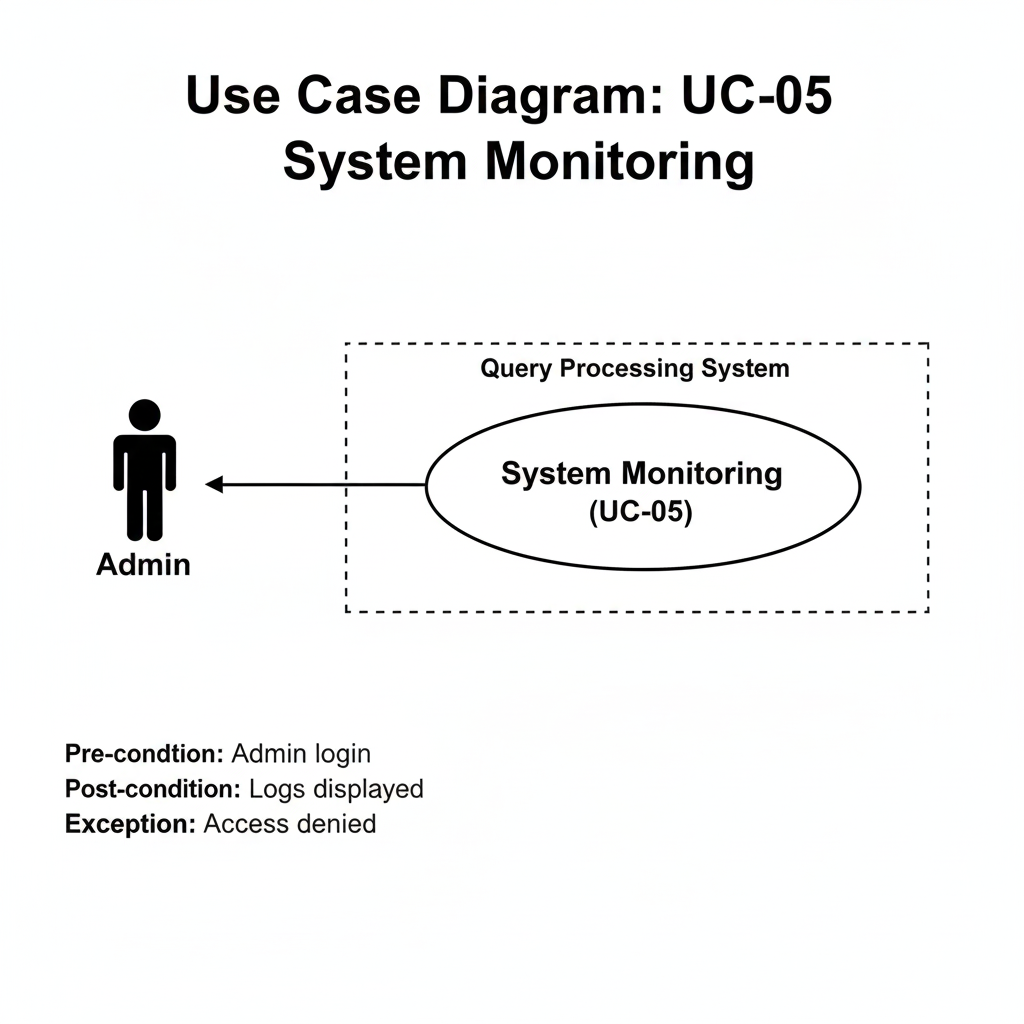


Figure3.6 Use Case for Admin Monitoring

|  |  |
| --- | --- |
| **Field** | **Description** |
| Use Case ID | UC-05 |
| Use Case Name | System Monitoring |
| Actor | Admin |
| Description | View logs and activity |
| Pre-condition | Admin login |
| Post-condition | Logs displayed |
| Exception | Access denied |

## 3.5 Summary

This chapter defined the complete system requirements for the Talk2SQL application. Functional requirements outlined the system’s capabilities in translating natural language queries into SQL, while non-functional requirements ensured security, performance, availability, and privacy. Use case diagrams and descriptions clarified system interactions for both users and administrators.

These requirements form the foundation for the system architecture and implementation, which are discussed in the next chapter.

# CHAPTER 4

## 4.0 Methodology

This chapter describes the methodology adopted for the development of **Talk2SQL**, an AI-powered system that converts natural language queries into structured SQL queries. The methodology defines the development approach, explains the rationale behind the selected model, and outlines the complete process flow followed during system design and implementation.

The goal of this methodology is to ensure that the system is developed in a structured, reliable, and iterative manner while maintaining flexibility to refine AI components such as natural language understanding, query generation, and database interaction.

## 4.1 Methodology for Software Development

Software development methodologies provide a structured framework that guides the planning, design, development, testing, and deployment of a system. For intelligent systems such as Talk2SQL, selecting an appropriate methodology is critical due to the involvement of AI models, database integration, and user interaction components.

Several software development methodologies were considered during the planning phase, including:

* Waterfall Model
* Agile Development Model
* Spiral Model
* Iterative Development Model

Each of these methodologies offers specific advantages. However, considering the academic nature of the project, the requirement for proper documentation, and the evolving behavior of AI components, a hybrid approach was required that could provide structure while allowing iterative refinement.

## 4.2 Selected Methodology

The **Iterative Waterfall Model** was selected for the development of the Talk2SQL system.

The Iterative Waterfall Model is a refined version of the traditional Waterfall Model, where development progresses through well-defined phases while allowing controlled iterations and feedback between phases. Unlike the classical waterfall approach, this model supports revisiting previous stages when improvements or corrections are required.

This model is particularly suitable for Talk2SQL because:

* The system involves **clearly defined stages** such as requirement analysis, NLP model development, query generation, and database execution.
* AI behavior can be refined iteratively based on testing outcomes.
* Documentation and academic evaluation remain well-structured.

4.1 Iterative Waterfall Model

A diagram of a software development process

Description automatically generated

## 4.3 Phases of the Iterative Waterfall Model

The development of Talk2SQL followed the phases outlined below:

### 4.3.1 Feasibility Study

In this phase, the technical and operational feasibility of Talk2SQL was evaluated. The study focused on determining whether natural language queries could be reliably transformed into SQL queries using modern NLP techniques and large language models.

Key outcomes of this phase included:

* Identification of supported SQL operations (SELECT, WHERE, JOIN, GROUP BY)
* Selection of AI models for query interpretation
* Assessment of database compatibility and execution constraints

### 4.3.2 Requirement Analysis

The requirement analysis phase involved gathering both functional and non-functional requirements for Talk2SQL.

Key requirements identified included:

* Acceptance of natural language queries from users
* Accurate translation of queries into SQL syntax
* Secure database connectivity
* Error handling for ambiguous or invalid queries
* Fast response time and scalability

This phase ensured that system expectations were clearly defined before moving to design and implementation.

### 4.3.3 System Design

The system design phase focused on defining the architecture and interaction between different components of Talk2SQL.

The system was divided into the following core modules:

* User Interface Module
* Natural Language Processing (NLP) Module
* SQL Query Generation Module
* Database Execution Module
* Response Formatting Module

Each module was designed to operate independently while maintaining seamless communication with other components.

### 4.3.4 Implementation

During the implementation phase, the system was developed according to the design specifications.

Key implementation activities included:

* Developing the frontend interface for user query input
* Integrating NLP and LLM-based models for query interpretation
* Designing SQL templates and query mapping logic
* Connecting the system to a relational database
* Executing generated SQL queries and fetching results

The system was implemented incrementally, allowing refinements after each iteration.

### 4.3.5 Testing

Testing was performed after each iteration to ensure correctness, accuracy, and reliability of the system.

Testing focused on:

* Validating SQL query correctness
* Handling ambiguous or incomplete user queries
* Performance testing for query execution
* Security testing for database access

Errors identified during testing were fed back into previous phases for refinement.

### 4.3.6 Deployment

After successful testing, the Talk2SQL system was deployed in a controlled environment. The deployment phase ensured that the system could be executed smoothly with real databases and handle user requests effectively.

### 4.3.7 Maintenance

The maintenance phase involves monitoring system performance and improving query accuracy over time. Enhancements to NLP understanding, support for additional SQL operations, and optimization of response time are handled during this phase.

## 4.4 Reason for Selecting the Iterative Waterfall Model

The Iterative Waterfall Model was selected for Talk2SQL due to the following reasons:

**4.4.1 Structured Development**

The model provides a clear sequence of phases, which is essential for academic documentation and evaluation.

**4.4.2 Controlled Flexibility**

Iterations allow refinement of AI components without disrupting the overall system structure.

**4.4.3 Risk Reduction**

Early testing of NLP and SQL generation reduces the risk of major failures in later stages.

**4.4.4 Ease of Documentation**

Each phase produces well-defined outputs, making it easier to document the system for an FYP report.

## 4.5 Brief Overview of Talk2SQL System Workflow

The Talk2SQL system follows a step-by-step workflow:

1. User enters a natural language query
2. NLP module interprets user intent
3. Query generation module converts intent into SQL
4. SQL query is validated and executed
5. Results are retrieved and formatted
6. Output is displayed to the user

### 4.5.1 System Design Overview

The system design ensures modularity, scalability, and maintainability. Each component is loosely coupled, allowing independent upgrades or improvements.

### 4.5.2 Development and Implementation Strategy

The development strategy focused on:

* Keeping the user interface simple and intuitive
* Ensuring SQL queries are syntactically correct
* Reducing hallucinations and incorrect query generation
* Supporting multiple database schemas

Iterative testing ensured continuous improvement in query accuracy.

## 4.6 Summary

This chapter presented the methodology used for developing the Talk2SQL system. The Iterative Waterfall Model was adopted to balance structured development with iterative refinement, making it suitable for an AI-driven academic project. Each development phase was clearly defined and executed to ensure reliability, scalability, and maintainability of the system.

The next chapter discusses the **architecture and design** of Talk2SQL, detailing system components, data flow, and database interaction mechanisms.

# CHAPTER 5

## 5.0 Architecture Design

This chapter presents the complete architectural design of the **Talk2SQL system**, explaining how different components interact to transform natural language queries into executable SQL statements and return meaningful results to the user. The architecture is designed to ensure **accuracy, scalability, security, and maintainability**, while supporting real-time interaction with a relational database.

The Talk2SQL system follows a **modular and layered architecture**, separating user interaction, natural language processing, query generation, database execution, and response formatting. This separation of concerns allows independent development, testing, and future enhancement of each module without disrupting the entire system.

This chapter covers the **activity flow**, **sequence of interactions**, **database design**, and the **overall system architecture**.

## 5.1 Activity Diagram

The activity diagram illustrates the high-level workflow of the Talk2SQL system, showing how a user’s natural language input is processed step by step until a final response is produced.

The process begins when a user enters a query in plain English (for example: *“Show total sales per product for last month”*). The system then validates the input, processes it using NLP techniques, generates an SQL query, executes it on the database, and returns the results in a human-readable format.

**Main Activities Involved**

* User authentication and authorization
* Natural language query input
* Query preprocessing and intent detection
* SQL query generation
* Query validation and execution
* Result formatting and visualization
* Error handling and feedback

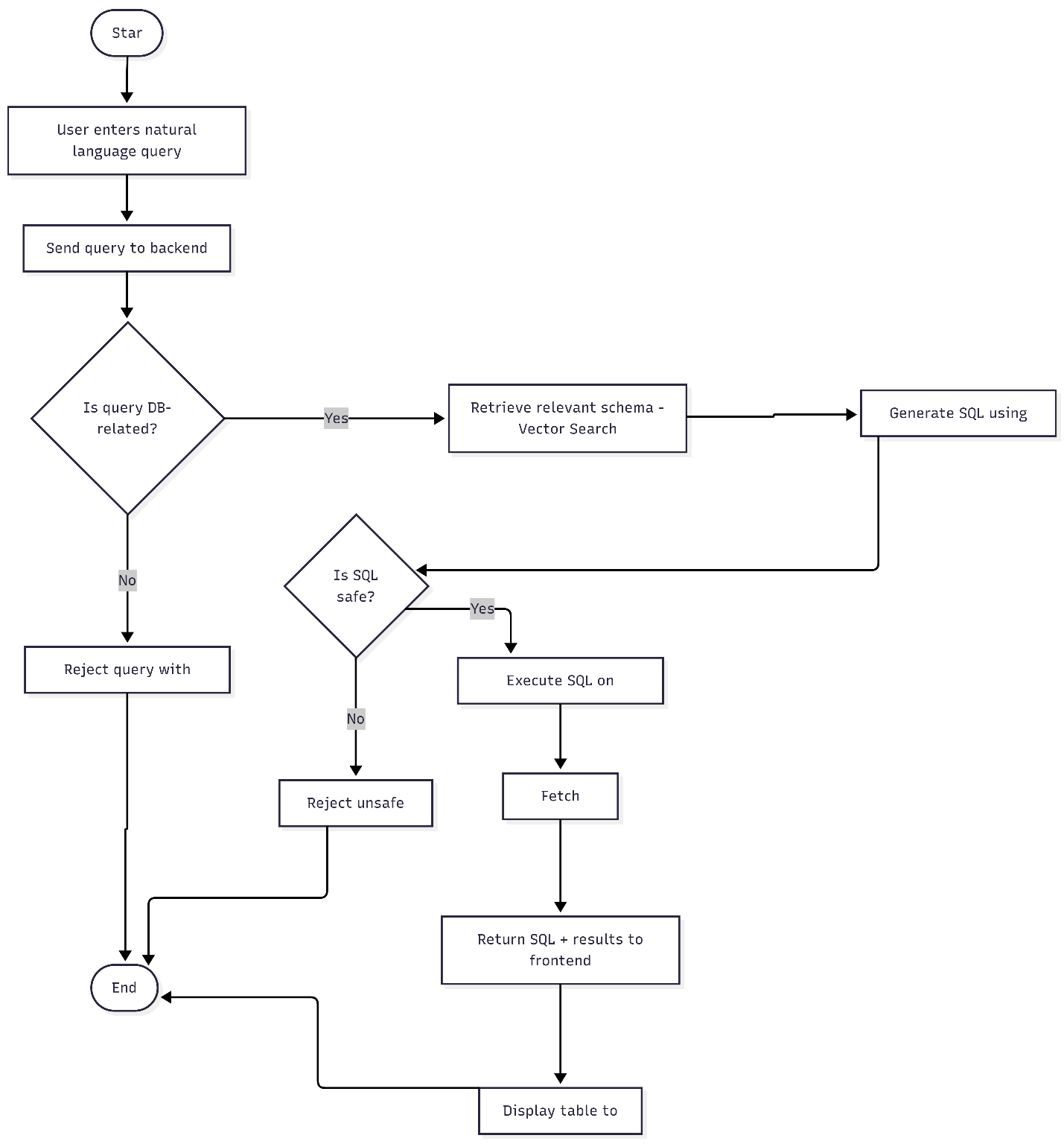


Figure 5.1 Activity Diagram of the Talk2SQL System

**Activity Flow Description**

1. **User Login:**  
   The user logs into the system based on their assigned role (user, analyst, or admin).
2. **Query Input:**  
   The user enters a natural language query related to employees, departments, inventory, production, or sales.
3. **Input Validation:**  
   The system checks for empty, ambiguous, or unsupported queries.
4. **NLP Processing:**  
   The query is tokenized, keywords are extracted, and intent is identified.
5. **Schema Mapping:**  
   Extracted entities are mapped to database tables and columns.
6. **SQL Query Generation:**  
   A valid SQL query is constructed dynamically.
7. **Execution:**  
   The query is executed on the relational database.
8. **Result Handling:**  
   The retrieved data is formatted into tables or summaries.
9. **Response Display:**  
   The final output is shown to the user.

### 5.1.2 Sequence Diagram

The sequence diagram illustrates the interaction between different components of the Talk2SQL system over time. It clearly shows how control flows from the user interface to backend services and finally to the database.

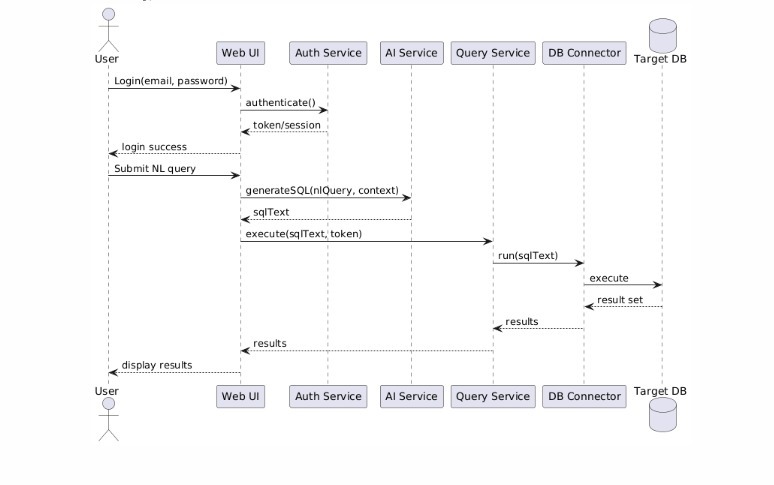


Figure 5.2 Sequence Diagram for Natural Language to SQL Execution

**Sequence of Interactions**

1. The **User Interface** sends the natural language query to the backend API.
2. The **NLP Engine** processes the query and extracts intent and entities.
3. The **Query Generator** converts the processed input into SQL.
4. The **Query Validator** checks for syntax and security constraints.
5. The **Database Engine** executes the SQL query.
6. The **Result Formatter** structures the output.
7. The **User Interface** displays the results.

This sequential interaction ensures controlled execution and prevents unauthorized or invalid database operations.

### 5.1.3 Database Design

The database design forms the backbone of the Talk2SQL system. A **relational database model** has been used, ensuring data consistency, integrity, and normalization up to **Third Normal Form (3NF)**.

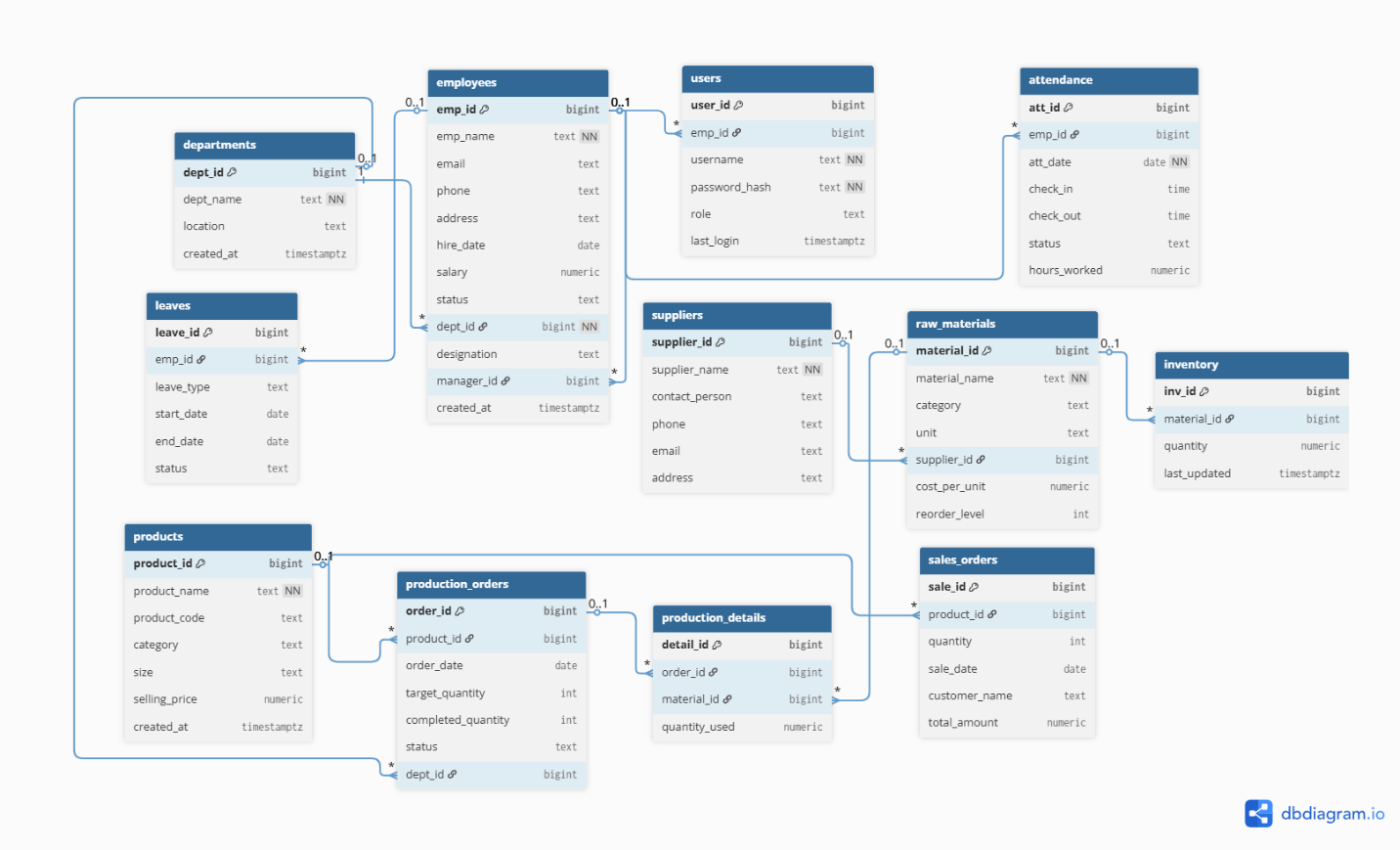
The schema supports core organizational operations including **employee management, attendance tracking, inventory control, production planning, and sales analysis**.

Figure 5.3 Database Design Diagram for Talk2SQL

### 5.1.4 Database Tables Overview

#### 5.1.4.1 Departments

Stores department-level information such as department name and location.

#### 5.1.4.2 Employees

Maintains employee details including department affiliation, salary, designation, and reporting hierarchy.

#### 5.1.4.3 Users

Handles system authentication and role-based access control.

#### 5.1.4.4 Attendance

Tracks daily attendance records for employees.

#### 5.1.4.5 Leaves

Manages employee leave requests and approval status.

#### 5.1.4.6 Suppliers

Stores supplier contact and identification information.

#### 5.1.4.7 Raw Materials

Contains information about materials used in production.

#### 5.1.4.8 Inventory

Tracks stock levels of raw materials.

#### 5.1.4.9 Products

Stores details of manufactured products.

#### 5.1.4.10 Production Orders

Manages production planning and progress.

#### 5.1.4.11 Production Details

Tracks material usage for each production order.

#### 5.1.4.12 Sales Orders

Stores sales transactions and revenue data.

### 5.1.5 Design Considerations

* Referential integrity enforced using foreign keys
* Cascading deletes for dependent records
* Role-based access through the users table
* Optimized query execution for analytics

### 5.1.6 System Architecture

The Talk2SQL system follows a **multi-tier architecture**, ensuring clear separation between presentation, processing, and data layers.

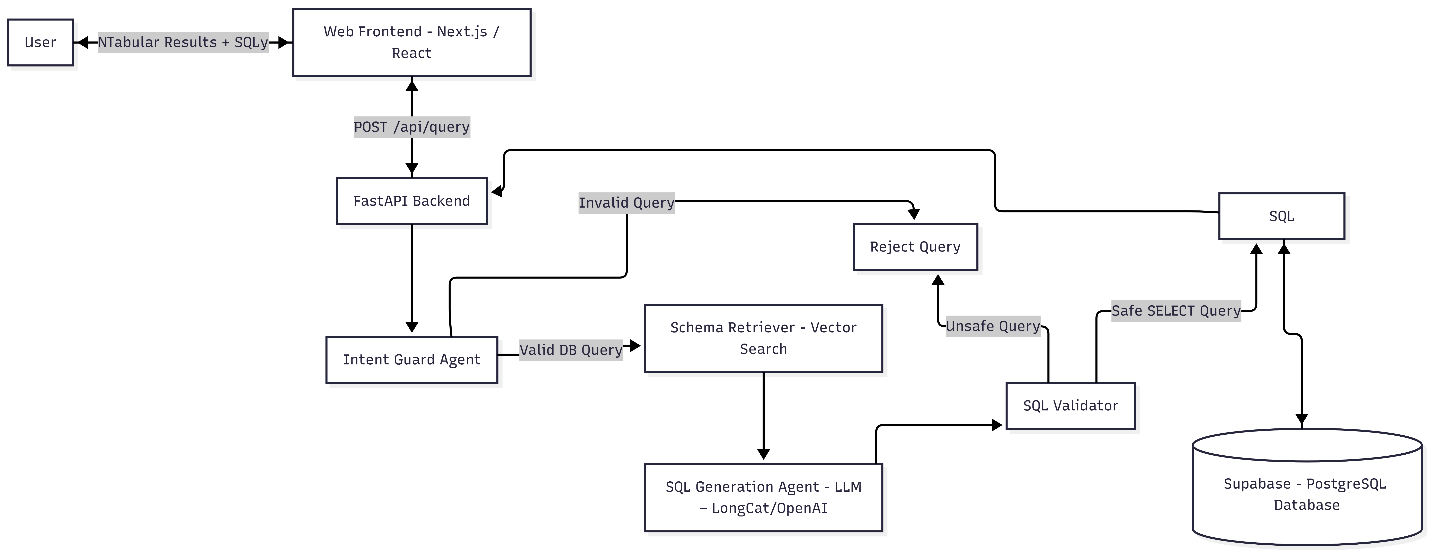


Figure 5.4 Overall System Architecture of Talk2SQL

### 5.1.7 Architectural Layers

#### 5.1.7.1 Presentation Layer

* Web-based user interface
* Accepts natural language queries
* Displays results and error messages

#### 5.1.7.2 Application Layer

* Handles authentication and authorization
* Controls request flow
* Communicates with NLP and query generation modules

#### 5.1.7.3 NLP & Query Processing Layer

* Natural language understanding
* Intent detection
* Schema mapping
* SQL query generation

#### 5.1.7.4 Database Layer

* Executes SQL queries
* Ensures data integrity and consistency
* Returns result sets

## 5.2 Summary

This chapter detailed the architectural design of the Talk2SQL system, explaining how user queries flow through different system components to generate accurate SQL queries and meaningful outputs. The activity and sequence diagrams highlighted the logical and temporal flow of operations, while the database design ensured data integrity and analytical capability.

The modular architecture supports scalability, maintainability, and security, making Talk2SQL a robust solution for natural language-based database interaction. This architectural foundation prepares the system for efficient implementation and testing, which are discussed in subsequent chapters.