CHAPTER 2
LITERATURE REVIEW

This study focuses on developing a natural language interface (NLI) for modern enterprise data infrastructure. In many organizations, timely operational decisions depend on data stored in an enterprise data warehouse. Access to that data, however, is often limited by technical barriers because non-technical users cannot always write SQL queries. This condition creates dependence on technical data teams and slows operational response.

This research positions a retrieval-augmented generation (RAG)-based conversational AI agent as an intermediary between operational teams and the enterprise data warehouse. Unlike conventional data-search systems, the AI agent approach emphasizes reasoning, context clarification, and query validation before an answer is returned. By using a large language model (LLM), the agent is designed to bridge complex enterprise database schemas and the need for fast, reliable, self-service data access.

2.1 Theoretical Foundation

2.1.1 Natural Language Interfaces to Data

Natural language interfaces to data (NLID) or natural language interfaces to databases (NLIDB) have been studied for decades. Androutsopoulos et al. (1995) introduced NLIDB as a system that enables users to ask questions in natural language, translates those questions into formal queries such as SQL, and returns the results in a form that is easier to understand. The study classifies early approaches into pattern-matching, syntax-based, and semantic-grammar approaches, while also noting limitations in portability to new domains and coverage of natural-language variation.

Quamar et al. (2022) expanded this perspective through a comprehensive survey of natural language interfaces to data. They describe the NLI processing chain from language understanding, including intent and entity recognition, to schema and data-source mapping, query generation, and result presentation. The

survey also discusses the shift from rule-based, statistical, and classical machine-learning approaches to deep learning and LLM-based systems, including the move from single-question interfaces to conversational systems that maintain context across turns.

Koutrika (2024) frames natural language data interfaces as part of a broader data-access journey, showing how the shift from formal query languages to natural-language and conversational systems changes the way users interact with data. One key point is that NLI quality is not determined only by translation accuracy into SQL, but also by how the system handles clarification, explanation, and collaboration with users when refining questions.

Conceptually, NLID theory provides the basis for designing an NLI architecture for BI. Functional blocks such as intent detection, entity recognition, data warehouse schema mapping, and SQL or analytical query generation become key elements that must be accommodated in the system design.

2.1.2 Conversational AI and Conversational Interfaces

Conversational AI is a branch of artificial intelligence that focuses on systems capable of interacting through natural language, either in text or voice. Bhagia et al. (2024) present a comprehensive survey of conversational AI, including system architecture, research trends, and common techniques. The survey shows that conversational AI architectures generally include automatic speech recognition (ASR) for voice input, natural language understanding (NLU), dialog management (DM), natural language generation (NLG), and text-to-speech (TTS) for voice output. Dialog management handles conversation flow, stores context, manages clarification, and connects the system to external services or data sources (Bhagia et al., 2024).

In the BI context, dialog management is important because analytical conversations often happen in stages, such as narrowing the data scope, choosing dimensions and metrics, and agreeing on the desired form of visualization.

The survey also emphasizes a shift from traditional modular systems to end-to-end deep learning approaches. However, modular systems remain useful when

high control and traceability over system behavior are required (Bhagia et al., 2024).

2.1.3 Natural Language Interfaces for Business Intelligence

Sura (2025) specifically reviewed natural language interfaces for business intelligence at enterprise scale. The article positions NLI as a potential solution for reducing the gap between business users and the complexity of data warehouses and analytical platforms. By using NLP and machine learning, NLI enables users to explore data, prepare reports, or ask analytical questions through simple conversational queries without needing to understand SQL or complex schemas. The review discusses the evolution, architecture, and experimental systems of NLI for BI, including GPT-4-based platforms. The main challenges highlighted include disambiguation of natural-language queries, scalability for large data volumes and complex enterprise schemas, explainability and transparency, security, governance, and regulatory compliance when NLI systems interact directly with critical organizational data (Sura, 2025).

These findings provide the theoretical basis for the system design in this thesis. NLI for BI is not only a natural-language understanding problem, but also a systems-engineering and data-governance problem in an enterprise environment.

2.1.4 LLM-Based Natural Language SQL Querying and Agent Architecture

Khan et al. (2025) studied natural language SQL querying using LangChain and an AI agent. Their prototype connects a web-based user interface, a large language model, and a SQL database. The architecture generally consists of a user interface where users enter natural-language questions, an LLM that interprets the question and generates SQL, a LangChain toolkit that connects the LLM to the database, and an execution mechanism that returns the query result as a narrative answer for the end user.

The study shows that LLM-based NL2SQL systems can reduce feature-engineering effort compared with traditional approaches because linguistic knowledge and common query patterns are already embedded in pretrained models. However, the authors also highlight hallucination, SQL syntax errors, and access-control risks when an LLM is allowed to execute queries directly against a database

(Khan et al., 2025). This concept is relevant to this thesis because an agentic approach allows the system to plan steps, choose tools, evaluate results, and improve queries, but it also requires clear validation mechanisms.

The behaviour that makes such an agent useful also explains why it has to be constrained. Jurafsky and Martin (2026) describe the large language model as a system trained by pretraining over large text corpora, so its output is produced by predicting a plausible continuation rather than by consulting a source of record. A model can therefore return a fluent statement that refers to a column or a table that does not exist in the schema. Mienye et al. (2025) reach a comparable conclusion from the architectural side. Their review traces foundational architectures from BERT and the GPT series to PaLM, LLaMA, and DeepSeek-R1, and proposes a taxonomy built on scalability, application domain, and ethical considerations, noting that interpretability and controllability remain open problems as model scale grows. Retrieval-augmented generation was introduced by Lewis et al. (2020) as one response to that problem, conditioning generation on documents retrieved for the specific input instead of on parametric memory alone, and Kamath et al. (2024) describe the resulting pattern, in which a retriever narrows the context before the generator produces an answer, as the standard way to tie model output to a controlled knowledge source. In this research the retrieved context is the semantic metadata of the warehouse, so the same mechanism is used to bound which tables and columns the model is allowed to reference.

Text-to-SQL research itself has moved through three stages that matter for this design. Encoder-decoder models first improved schema linking by representing table and column relationships explicitly, as in the relation-aware schema encoding of Wang et al. (2020). Constrained decoding then addressed syntactic validity, and Scholak et al. (2021) reject partial outputs that cannot lead to a well-formed query while the query is still being generated. With large language models the emphasis shifted to prompt structure and self-correction, and Pourreza and Rafiei (2023) added an explicit schema-linking step with difficulty-aware instructions before generation. Su et al. (2026) argue that this line of work is still measured mainly through string-level comparison with a gold query, which can raise exact-match scores without guaranteeing that the query executes, and they separate schema

selection, structure-aware generation, and complexity-aware refinement into three stages. Li et al. (2024) reach a comparable cautionary conclusion when they assess whether natural language to SQL is ready for practical use.

Two studies sit closest to the setting of this research. Zhao et al. (2024) combine retrieval-augmented generation, a vector database, and a large language model into an interactive data-analysis system, which is the same combination used here for metadata retrieval. Du et al. (2025) adapt natural language to SQL for the financial industry and show that domain-specific processing has to be built around the model rather than expected from it. Neither study reports a deterministic security layer placed between the generated query and the database, and that gap is what the present research addresses.

2.1.5 Evaluation Metrics for Natural Language Interfaces to Data

Evaluating a natural-language interface to a data warehouse requires more than one kind of measure, because a system can produce syntactically valid SQL and still return the wrong figure, answer too slowly to be useful, or fail to earn the trust of the people who rely on it. Baig et al. (2025) group the measures used in the Text-to-SQL literature into query-level correctness, execution correctness, and dialog-level correctness, and note that operational studies usually add performance and acceptance measures on top of them. This study follows that grouping, so the metrics applied in Chapter 3 and reported in Chapter 4 are defined here first.

Query-level correctness is measured by exact match (EM), which is the proportion of predicted queries that are structurally identical to the reference query. Execution correctness is measured by execution accuracy (EX), which is the proportion of predicted queries whose result set equals the result set of the reference query after both are executed. Dialog-level correctness is measured by interaction match (IM), which counts a multi-turn session as correct only when every question in that session is answered correctly. For a set of n evaluated items, each of the three is computed as the number of items satisfying the criterion divided by n.

Performance is measured through response latency. For scenario s, web latency is the interval between the moment the question is submitted in the interface and the moment the answer is rendered, written as L_web(s) = t_rendered(s) -

t_submitted(s), while backend latency L_api(s) is the processing time reported by the API for the same request. The distribution over n scenarios is summarized by its minimum, median, and mean, where the mean is the sum of L(s) over all s divided by n. Operational usefulness is measured through task-completion efficiency, defined as TCE = (c / n) x 100 percent, where c is the number of scenarios that reach the expected terminal state, either answered or refused, without any manual SQL writing or correction, and n is the total number of scenarios.

Usability is measured with a five-point Likert instrument. Let x_ir denote the score for evaluation item i from respondent r, where i ranges from 1 to k items and r ranges from 1 to m respondents. In Equations (1) to (4), A_i represents the average score for item i, R_r represents the mean score for respondent r, M represents the overall mean, and P represents the percentage of the maximum possible score. Rounding is applied only to the final reported values. These four equations are used to aggregate the usability scores reported in Section 4.4.7.

$$A_i = \frac{x_{i1} + x_{i2}}{2} \quad (1)$$

$$R_r = \frac{\sum_{i=1}^{7} x_{ir}}{7} \quad (2)$$

$$M = \frac{\sum_{r=1}^{2} \sum_{i=1}^{7} x_{ir}}{14} \quad (3)$$

$$P = \frac{M}{5} \times 100\% \quad (4)$$

2.2 Previous Studies

This section is written as a critical review. Each study is discussed in terms of its contribution, relevant limitations, and implications for this research design. A summary of the reviewed studies is presented in Table 2.1.

Table 2.1 Previous Studies

No.	Author(s)	Focus and Main Findings	Relevant Limitations	Implications for This Research
1	Sura (2025)	Reviews natural-language interfaces for business intelligence at enterprise scale, with emphasis on disambiguation, scalability,	The discussion is primarily conceptual and does not provide a complete implementation of deterministic SQL validation and output masking.	Supports the need for a controlled architecture that combines semantic retrieval with validation before query execution.

No.	Author(s)	Focus and Main Findings	Relevant Limitations	Implications for This Research
		explainability, security, and governance.		
2	Quamar et al. (2022)	Surveys the complete natural-language interface pipeline, including intent interpretation, schema mapping, query generation, and result presentation.	Large and changing schemas make context selection and schema linking difficult.	Supports retrieval of a limited, relevant metadata context before SQL generation.
3	Koutrika (2024)	Argues that interface quality depends on clarification, collaboration, and explanation, not only translation accuracy.	Secure database execution is not the main focus.	Supports conversation context, clarification, SQL transparency, and understandable result presentation.
4	Androutsopoulos et al. (1995)	Establishes the classical natural-language interface process of translating a user question into a formal database query.	Classical systems were constrained by narrow domains and limited linguistic coverage.	Provides the conceptual foundation for using modern language models while retaining explicit database controls.
5	Bhagia et al. (2024)	Reviews conversational AI architectures, dialog management, and context handling.	The study is not specific to enterprise BI or Text-to-SQL governance.	Supports multi-turn session management and clarification before query execution.
6	Ponnusamy (2023)	Describes the development of enterprise data warehouses and their role in analytical decision support.	Does not examine natural-language querying in depth.	Establishes the data-warehouse context and the need to simplify access for non-technical users.
7	Kobeissi et al. (2021)	Uses intent-based interpretation to query process-execution data and improve control over intent and slot mapping.	The evaluated process-data domain is narrower and more homogeneous than an enterprise data warehouse.	Supports explicit intent and entity interpretation before metadata selection and query planning.

No.	Author(s)	Focus and Main Findings	Relevant Limitations	Implications for This Research
8	Khan et al. (2025)	Implements an agent-based natural-language-to-SQL workflow using LangChain, an LLM, query execution, and answer presentation.	The prototype gives limited attention to deterministic enterprise security controls and broad evaluation.	Provides an agent-workflow reference that this research extends with metadata restrictions, SQL validation, read-only execution, and masking.
9	Wang et al. (2020)	Introduces relation-aware schema encoding and linking for text-to-SQL parsers, representing table and column relationships and foreign-key dependencies explicitly so that the decoder receives richer schema context.	The approach depends on a trained parser for a fixed schema representation and does not address access control or execution safety.	Confirms that schema context must be selected and structured before generation, which this research implements through semantic metadata retrieval.
10	Scholak et al. (2021)	Applies constrained decoding, rejecting partial outputs that cannot lead to a well-formed query while the SQL is still being generated, which guarantees syntactic validity.	Constrained decoding enforces form rather than intent, so a query can be syntactically valid yet inconsistent with what the user asked for.	Supports validating the statement before execution, and shows why validation alone is not sufficient without a semantic layer.
11	Pourreza & Rafiei (2023)	Decomposes in-context learning for text-to-SQL by adding an explicit schema-linking step and difficulty-aware instructions to the prompt, together with a self-correction stage.	The model is still treated as a standalone generator, so results remain sensitive to prompt design and to question complexity.	Provides the basis for the prompt-construction stage used in this prototype, which supplies only retrieved metadata as the allowed schema.
12	Zhao et al. (2024)	Builds an interactive data-analysis system	The system focuses on analytical interaction and does	Directly supports the retrieval design used here, where metadata

No.	Author(s)	Focus and Main Findings	Relevant Limitations	Implications for This Research
		that combines retrieval-augmented generation, a vector database, and a large language model.	not report a deterministic security layer between the generated query and the database.	embeddings are stored and searched with pgvector.
13	Li et al. (2024)	Assesses whether natural language to SQL is ready for practical deployment and reviews the gap between benchmark results and real database use.	The assessment is diagnostic and does not propose an operational control architecture.	Justifies reporting the evaluation of this prototype as controlled evidence rather than as a generalizable accuracy claim.
14	Du et al. (2025)	Adapts natural language to SQL for the financial industry by combining Python-based processing with a large language model to meet domain-specific requirements.	The work is domain-specific and does not cover sensitive-data masking or read-only execution policy.	Shows that enterprise deployment needs domain processing built around the model, which this research provides through business metadata and curated views.
15	Mienye et al. (2025)	Reviews foundational large language model architectures from BERT and the GPT series to PaLM, LLaMA, and DeepSeek-R1, and proposes a taxonomy based on scalability, application domain, and ethical considerations.	The review remains at the architectural level and does not evaluate model behaviour against a live database schema or an access-control policy.	Supports keeping interpretability and access control outside the model instead of relying on model behaviour alone.
16	Su et al. (2026)	Proposes a three-stage text-to-SQL framework that separates schema selection, structure-aware generation, and	Evaluation is performed on public benchmarks rather than an enterprise warehouse, and the framework does not address data-leakage	Supports separating retrieval, generation, and validation into distinct stages, which is the structure adopted by the prototype in this research.

No.	Author(s)	Focus and Main Findings	Relevant Limitations	Implications for This Research
		complexity-aware refinement, and reports state-of-the-art execution accuracy on the Spider benchmark.	prevention.	

Based on the previous studies, the research gaps addressed in this thesis are as follows:

1) There is a need for an end-to-end NLI for BI design that not only generates SQL, but also provides execution control and auditability.
2) The need for an effective context selection mechanism on a large warehouse schema
3) The need to combine LLM flexibility with consistent control interpretations and metric definitions.

Therefore, this study positions RAG, the SQL agent, and query validation as a controlled workflow for enterprise data warehouse access.