SiriusBI: A Comprehensive LLM-Powered Solution for Data Analytics in Business Intelligence

Jie Jiang[1], Haining Xie[1], Siqi Shen[2], Yu Shen[1], Zihan Zhang[1], Meng Lei[1], Yifeng Zheng[1], Yang Li[1], Chunyou Li[1], Danqing Huang[1], Yinjun Wu[3], Wentao Zhang[2], Bin Cui[3], Peng Chen[1]
[1]Department of Data Platform, TEG, Tencent Inc.
[2]Center of Machine Learning Research, Peking University
[3]School of Computer Science, Peking University
[1]{zeus, hainingxie, willyushen, rylanzhang, garylei, yifengzheng, thomasyngli, chunyouli, daisyqhuang, felixxfyang, pengchen}@tencent.com
[2]{shensiqi1009, wentao.zhang}@pku.edu.cn [3]{wuyinjun, bin.cui}@pku.edu.cn

ABSTRACT

With the proliferation of Large Language Models (LLMs) in Business Intelligence (BI), existing solutions face critical challenges in industrial deployments: functionality deficiencies from legacy systems failing to meet evolving LLM-era user demands, interaction limitations from single-round SQL generation paradigms inadequate for multi-round clarification, and cost for domain adaptation arising from cross-domain methods migration.

We present SiriusBI, a practical LLM-powered BI system addressing the challenges of industrial deployments through three key innovations: (a) An end-to-end architecture integrating multi-module coordination to overcome functionality gaps in legacy systems; (b) A multi-round dialogue with querying mechanism, consisting of semantic completion, knowledge-guided clarification, and proactive querying processes, to resolve interaction constraints in SQL generation; (c) A data-conditioned SQL generation method selection strategy that supports both an efficient one-step Fine-Tuning approach and a two-step method leveraging Semantic Intermediate Representation for low-cost cross-domain applications. Experiments on both real-world datasets and public benchmarks demonstrate the effectiveness of SiriusBI. User studies further confirm that SiriusBI enhances both productivity and user experience.

As an independent service on Tencent's data platform, SiriusBI is deployed across finance, advertising, and cloud sectors, serving dozens of enterprise clients. It achieves over 93% accuracy in SQL generation and reduces data analysts' query time from minutes to seconds in real-world applications.

PVLDB Reference Format:
Jie Jiang, Haining Xie, Siqi Shen, Yu Shen, Zihan Zhang, Meng Lei, Yifeng Zheng, Yang Li, Chunyou Li, Danqing Huang, Yinjun Wu, Wentao Zhang, Bin Cui, Peng Chen. SiriusBI. PVLDB, 18(12): 4860 - 4873, 2025.
doi:10.14778/3750601.3750610

PVLDB Artifact Availability:
The source code, data, and/or other artifacts have been made available at https://github.com/Tencent-SiriusAI/SiriusBI.

Proceedings of the VLDB Endowment, Vol. 18, No. 12 ISSN 2150-8097.
doi:10.14778/3750601.3750610

1 INTRODUCTION

Business Intelligence (BI) [54, 83] is a crucial application scenario in the data field, comprising a comprehensive suite of methodologies, tools, and infrastructures designed to collect, integrate, analyze, and present raw data from an organization to generate actionable insights for informed decision-making. BI systems are extensively used in various sectors, including finance [55], environment [24], and social media [11, 64], which significantly improves the decision-making process through the provision of real-time analytics and reporting capabilities [44, 60].

A typical BI system comprises several key components: a data management module that stores, processes, and aggregates vast amounts of data; analytic algorithms that transform the data into actionable insights; and visualization tools that present the information in intuitive and user-friendly formats. Among these, data analytics plays a crucial role in providing decision-making support, directly determining the correctness and appropriateness of decisions. Recent advancements in LLMs [34, 46, 89] have sparked significant interest in ChatBI — a new paradigm supported by natural language interfaces [1, 41]. Concurrently, the demand for a fully integrated and efficient ChatBI solution is surging, driven by the need of a more intuitive and accessible mode of data interaction. This evolution promises to transform how users engage with data, making insights more available and actionable.

To meet the growing demand for big data analytics and decision-making in BI, the data community has proposed numerous effective approaches. However, when applying existing work in real-world BI scenarios, we identify the following three challenges:

C1: Functionality Deficiencies. While traditional business intelligence systems [8] integrate core components spanning data management, SQL generation, and insight discovery to form complete analytics pipelines, their reliance on heuristic rules and conventional AI/ML techniques limits generalization ability in dynamic scenarios. Although LLM-based methods have advanced task-specific performance, few offer comprehensive BI capabilities comparable to their traditional counterparts. For example, MAC-SQL [75] and CHESS [68] optimize NL2SQL accuracy but treat SQL execution as terminal outputs, neglecting downstream tasks like attribution analysis. While Lian et al. [46] extend their pipeline with Apache Superset for visualization, they fail to introduce knowledge bases to support dynamic grounding of domain-specific context, a

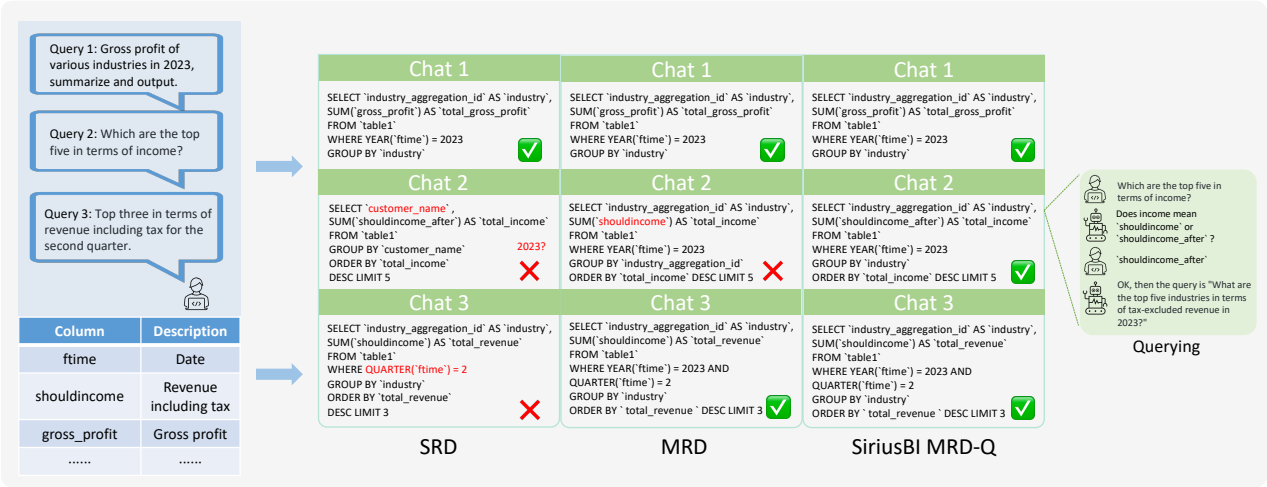


Figure 1: Demonstration of multi-round user requests. Compared with SiriusBI, SRD misses the omitted year information in conversations, while MRD fails to identify the user's ambiguous intent.

critical requirement for real-world BI adaptation [12]. This functional fragmentation forces users to manually coordinate tools (e.g., SQL editors, dashboard platforms, knowledge retriever), which imposes significant cognitive load and reduces operational efficiency, as evidenced by industry report [36].

C2: Interaction Limitations. In the context of ChatBI, the NL2SQL task is becoming increasingly vital, as it facilitates seamless interaction between natural language queries and structured data retrieval, thereby enhancing the efficiency and accuracy of data analytics. The evolution of NL2SQL techniques reveals a critical architectural mismatch: while traditional methods (schema-based [25, 26, 50, 87] or parsing-based [30, 40, 56, 77]) and modern LLM-driven approaches (prompt engineering [58, 75] or fine-tuning techniques [42, 59]) predominantly optimize for **S**ingle-**R**ound **D**ialogue (**SRD**) precision. This SRD-centric paradigm introduces a significant continuity gap in **M**ulti-**R**ound **D**ialogues (**MRD**): real-world BI workflows often require iterative investigation through successive queries, where later queries tend to omit previously provided contextual information, resulting in semantic ambiguity beyond the initial query. For instance, in the MRD NL2SQL task illustrated in Figure 1, the user issues three queries; notably, the second and third queries omit the time condition "2023" because it was specified in the first query. Single-round NL2SQL approaches demand nearly perfect input specificity, which explains its failure to generate correct SQL statements for the second and third queries in Figure 1. Worse still, due to the intricate nature [48] of MRD, few approach has been devoted to addressing this task. Lian et al.'s MRD solution [46] is the first attempt towards this task. Nevertheless, their solution is absent of user-guided clarification loops for intent resolution and domain-grounded dialogue act modeling. As demonstrated in Figure 1, the basic MRD approach still exhibit performance degradation beyond the first dialogue round.

C3: Cost for Domain Adaptation. Cross-domain deployment of NL2SQL models faces the challenge of cost surges in domain knowledge transfer, primarily caused by insufficient model generalization capability. Structural differences in database schema across domains (e.g., nested tables in finance vs. wide tables in advertising) necessitate repetitive model adaptation [48], while semantic gaps between industry-specific operators (e.g., financial window functions vs. e-commerce promotional rules) exacerbate logical deviations in SQL generation [38]. Critically, domain knowledge transfer relies heavily on expert-annotated data, with manual annotation costs growing with domain complexity [39]. Our real-world deployment statistics show that direct model migration leads to business logic errors in approximately two-thirds of generated SQL queries. Meanwhile, adapting models through traditional fine-tuning requires 5.5 person-days on average to label 200 seed queries within existing databases—forming critical bottlenecks for enterprise-level scalability.

To address the aforementioned challenges, we propose SiriusBI, which implements a comprehensive LLM-powered solution for ChatBI scenarios. This system leverages the capabilities of LLMs to empower various modules, thereby enhancing both the efficiency and user experience in data analytics. Specifically, for the issue of functionality deficiencies (**C1**), SiriusBI introduces an **end-to-end** integrated architecture that seamlessly orchestrate core modules including knowledge management, multi-round dialogue analysis, SQL generation, and data insight provision, thereby ensuring a closed-loop pipeline from natural language queries to final decision-making reports.

For the issue of interaction limitations (**C2**), we introduce the **MRD-Q** (**M**ulti-**R**ound **D**ialogue with **Q**uerying) module. As a supplement to the basic multi-round dialogue analysis module proposed by Lian et al. [46], MRD-Q incorporates an intent querying module to clarify user queries through follow-up questions. This approach enables the system to accurately identify the user's true intent, even when the initial query is incomplete or ambiguous, thus facilitating precise responses, as shown in Figure 1.

To enable economic domain adaptation (**C3**), SiriusBI introduces a strategy switching mechanism that dynamically selects between one-step and two-step SQL generation paradigms based on data conditions. This mechanism optimizes the trade-off between computational cost and performance, ensuring efficient adaptation to

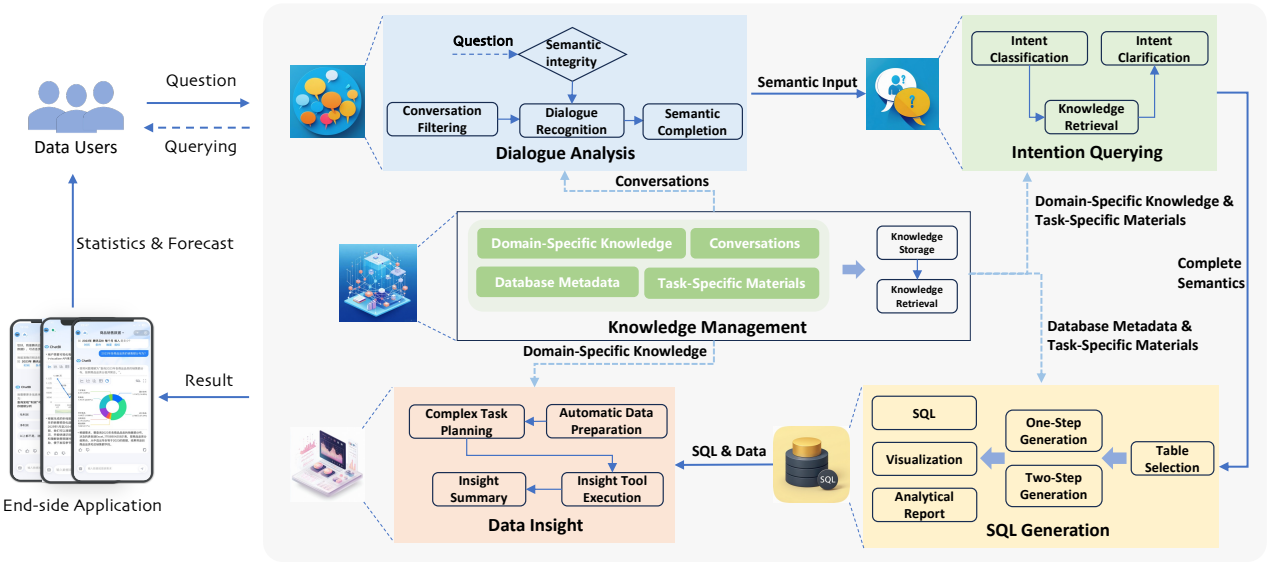


Figure 2: Workflow overview of SiriusBI.

diverse business domains. To address structural and semantic disparities across domains, the one-step method employs a supervised fine-tuning (SFT) method enhanced by an automated data generation process pipeline. This pipeline minimizes manual annotation costs by leveraging SQL logs and LLM-based reverse engineering to synthesize high-quality training data, greatly reducing reliance on labor-intensive expert labeling. For cold-start scenarios, the two-step method eliminates SFT by combining domain knowledge retrieval with semantic intermediate representation (SIR), achieving robust performance across diverse industries without domain-specific tuning. The generalization ability of our system is further enhanced by a knowledge base with hybrid storage strategies, specially designed for BI application scenario.

Contributions. The main contributions of this paper can be summarized as follows:

- Practical ChatBI System: SiriusBI integrates four core functionalities — knowledge management, multi-round dialogue with querying, SQL generation, and data insight — into a unified system. It has been deployed across Tencent's finance, advertising, and cloud businesses departments, serving dozens of enterprise clients, reducing data analysis latency from minutes to seconds.
- New Dialogue Analysis Mechanism: We propose a new dialogue analysis mechanism named MRD-Q, consisting of semantic completion, knowledge-guided clarification, and proactive querying processes. It effectively addresses issues raised by ambiguous or incomplete queries in real-world scenarios.
- Economic Domain Adaptation Strategy: We propose a dynamic strategy selection mechanism for SQL generation conditioned on data characteristics. To reduce annotation costs for fine-tuning, we set up an automated data synthesis pipeline. To enhance cross-domain SQL syntax accuracy and business logic consistency, we use Semantic Intermediate Representation (SIR) in prompt engineering.
- Multi-Round NL2SQL Benchmark: We construct *MRD-BIRD* —a dataset containing 96 multi-round dialogue sequences (3–5 interaction rounds)— and open-source the data to advance NL2SQL research in multi-round dialogue contexts.
- Proven Effectiveness: Extensive experiments on both academic benchmarks and real-world datasets demonstrate the effectiveness of SiriusBI. Notably, when applied in industry, SiriusBI achieves remarkable accuracy rates of **97%** in SQL generation for Tencent Finance, **93%** for Tencent Advertisement, and **96%** for Tencent Cloud. User studies further confirm that SiriusBI enhances both productivity ad user experience.

2 RELATED WORK

2.1 Business Intelligence

The integration of AI and ML has significantly advanced data-driven decision-making in BI [62], with predictive analytics enhancing forecasting and pattern recognition [37, 69], and NLP improving the extraction of insights from unstructured data [29, 33]. The emergence of the ChatBI paradigm [47], enabled by powerful models such as BERT [15] and ChatGPT [72], has further promoted data democratization through conversational interfaces and intelligent data interpretation. These systems allow non-technical users to interact with data assets using natural language, significantly lowering the barrier to business analytics. BI-REC [52] is an early system that recommends BI applications by predicting user intents based on historical interactions. In contrast, our approach focuses on assessing whether the current user intent is sufficiently clear for accurate SQL generation. Lian et al. [46] are the first to emphasize the NL2BI task in ChatBI and propose a lightweight method for generating semantically complete queries. Building on their insights, we (1) adopt LLMs to enhance query completeness assessment, (2) inject domain-specific knowledge to improve intent understanding, and (3) introduce user-facing clarification when ambiguity is detected—two capabilities not addressed in their work.

2.2 NL2SQL Methods

NL2SQL is a core task in ChatBI systems and can be broadly categorized into prompt-based and training-based methods [91]. Recently, Large Language Models (LLMs) like Codex [73] and Claude 3 [70] have shown great promise in NL2SQL, leveraging techniques such as prompt engineering [23], in-context learning [16], and chain-of-thought prompting [81]. Prompt-based approaches not only offer new technical paradigms but also reduce costs in industrial settings. Notable examples include DIN-SQL [58], C3 [17], MAC-SQL [75], and SQL-PaLM [66], which improve NL2SQL performance on models like ChatGPT and PaLM [53].

Before LLMs, NL2SQL solutions mainly relied on training encoder-decoder sequence-to-sequence models [6, 7, 13, 20, 76], requiring large amounts of database and SQL data. The advent of LLMs has introduced new training paradigms; for instance, Codes [42] proposes a training method tailored for NL2SQL that boosts LLM performance. However, some studies [5, 51, 57] fine-tune LLMs like ChatGPT [72] and Gemini [71] to further improve results, though this often incurs high practical costs.

3 METHOD

3.1 Overview

The overall framework of SiriusBI is illustrated in Figure 2 and consists of four core modules:

(1) The **Knowledge Management** module consists of four types of data – database metadata, domain-specific knowledge, task-related materials (e.g., demonstrations used for in-context learning in LLMs), and historical conversations. These data types are stored in a hybrid architecture, ensuring efficient and scalable access to knowledge, supporting the seamless functionality of other modules.

(2) The **Multi-Round Dialogue with Querying (MRD-Q)** module is composed of two components: *Dialogue Analysis* and *Intention Querying*, which enable multi-turn user-oriented dialogue and knowledge-guided clarification to resolve semantic ambiguity in Conversational BI. Unlike existing SRD-centric approaches [58, 75], MRD-Q introduces follow-up question generation based on domain-specific knowledge, enabling context-aware SQL calibration across 3-5 dialogue turns.

(3) The **SQL Generation** module dynamically selects between one-step (fine-tuning based) and two-step (semantic intermediate representation based) processing paradigms according to business scenario data characteristics. This dual-mode approach efficiently transforms user intent into SQL queries with optimal cost-effectiveness. The one-step method streamlines the process through an automated data preparation pipeline, while the two-step method employs a knowledge-base-driven query rewriting strategy that eliminates model training requirements. This architectural design balances operational efficiency with implementation flexibility across diverse application contexts.

(4) To close the loop from SQL execution to decision support, SiriusBI integrates a **Data Insight** module that interprets query results through a multi-agent workflow based on the FunctionCall [35] and ReAct [86] mechanism as shown in Figure 3. Specifically, the *Planner Agent* analyzes user queries, SQL results, and domain knowledge to decompose tasks into subtasks and issue instructions; the *Data Preparation Agent* identifies missing data requirements and generates SQL queries to retrieve necessary information; the *Tool Execution Agent* selects specialized tools (Forecast/Diagnosis/Attribution Tools, implemented via reproduced existing works [2, 3, 28, 45, 49]) to process data, with final outputs consolidated by the Planner Agent for response generation.

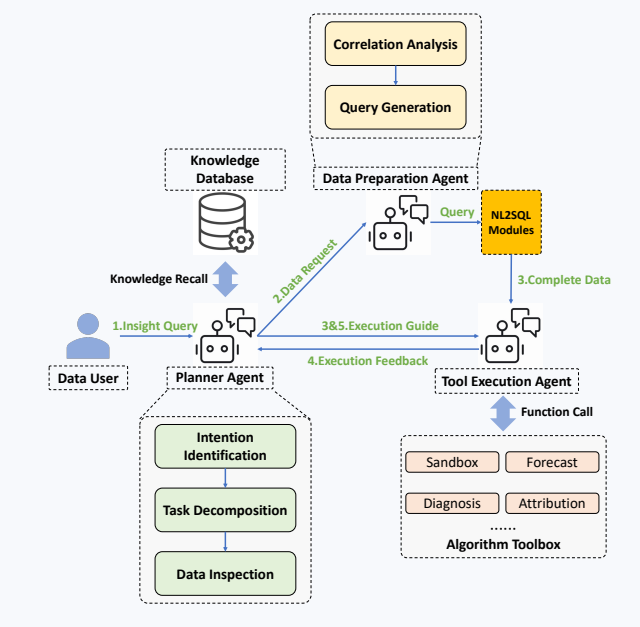


Figure 3: Detailed description of the task planning workflow for SiriusBI data insights in response to user queries.

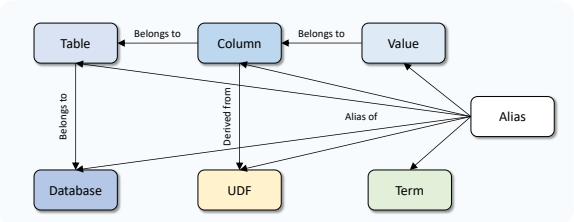


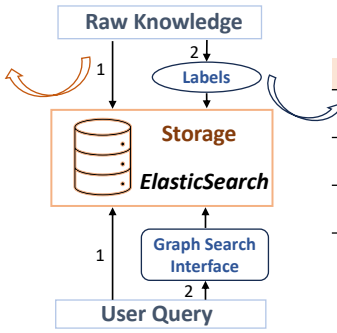
Figure 4: The hierarchical structure of database schema and some related domain-specific knowledge.

3.2 Knowledge Management

The application of BI often requires substantial amount of domain-specific knowledge. To address the knowledge requirements, SiriusBI constructs and maintains a comprehensive knowledge base within the knowledge management module. This knowledge base is designed to support efficient data retrieval, enhance decision-making processes, and facilitate seamless integration with BI tools.

The knowledge base primarily constitutes four types of data: (1) Database Metadata: This includes essential information such as table names, column types, column names, and other details of the database schema. (2) Domain-Specific Knowledge: This encompasses explanations of certain fields, term definitions, and relevant

Name	Description	Demonstration
Product Name	The corresponding field is `product_name`	How many products are there in the product name that contains "smart"?
Smart Speaker	Belong to the product name (`product_name`)	/
School Recruitment	Belong to the recruitment path (`recruit_channel`)	How many people have joined through school recruitment this year?
Company	By default, without special instructions, refer to Tencent	/



Name	Label	Anc. Name	Anc. Label
Personnel Table	Table	Company HR Database	Database
Recruitment Channel	Column	Personnel Table	Table
Campus Recruitment	Value	Recruitment Channel	Column
Campus Hiring	Alias	Campus Recruitment	Value

Figure 5: Illustration of the hybrid storage strategy.

business context. It provides the necessary background information to interpret and utilize the data effectively within the specific domain. (3) Task-Specific Materials: These are groups of prompts, demonstrations and other resources designed for different tasks. They are tailored to guide users in performing specific operations or analyses within the BI system. (4) Historical Conversations: This includes logs of past interactions and queries, which can be used to complete semantic for multi-round dialogue analysis.

3.2.1 Knowledge Storage. We observe that certain types of knowledge entries exhibit dependencies, while others remain relatively independent. In our application scenario, the relationships between database metadata and relevant domain-specific knowledge can be represented as a directed acyclic graph as illustrated in Figure 4. To ensure comprehensive retrieval of all related knowledge, it is essential to explicitly capture and represent these relationships.

While graph databases [21] offer a natural solution for storing such relational knowledge, we opt against their use due to their inherent complexity and overhead, which exceed the requirements of our system. Instead, we adopt a lightweight approach by augmenting knowledge entries with two dedicated fields: `Anc. Label`, which specifies the label of the ancestor entry, and `Anc. Name`, which encodes the concrete value of the ancestor entry. This design, as depicted in Figure 5 (right table), preserves the relational semantics necessary for knowledge retrieval while avoiding the operational and computational complexities associated with native graph storage. This approach strikes a balance between functionality and efficiency, aligning with the practical needs of our system.

For independent knowledge entries, we simply store a description of the entry and, if applicable, a demonstration or example. This straightforward approach is sufficient for entries that do not have dependencies or relationships with other knowledge items, as shown in Figure 5 (left table).

3.2.2 Knowledge Extraction. The objective of the knowledge extraction process in SiriusBI is to efficiently and accurately extract relevant information from the knowledge base. We employ different knowledge extraction strategies based on the storage strategy described earlier. All queries must undergo at least one round of knowledge retrieval, which is divided into two main phases: coarse retrieval and fine retrieval. This two-phase approach ensures a balance between recall and precision, enabling the system to first identify a broad set of potentially relevant knowledge and then refine the results to meet the specific needs of the task at hand.

In the coarse retrieval phase, the primary goal is to maximize recall, ensuring that a comprehensive set of potentially relevant knowledge entries is retrieved. This is particularly important for tasks such as SQL generation, where the system needs to identify all possible column names or metadata that might be relevant to the user's query. To achieve this, we employ a mixed-metric search strategy that operates on both lexical and semantic levels [78]. Lexical search captures exact matches or close variants of the query terms, while semantic search leverages embeddings to identify conceptually related knowledge entries, even if they do not share exact lexical overlap. For this phase, we utilize Elasticsearch [18], a widely adopted search engine known for its scalability and efficiency. The query is processed without segmentation, allowing for a broader initial recall set.

For knowledge entries that exhibit hierarchical relationships (e.g., those with `Anc. Name` fields), we implement a multi-round retrieval strategy inspired by multi-hop graph traversal. Specifically, for each entry retrieved in the initial search, if its `Anc. Name` field is non-empty, we recursively retrieve the ancestor entries. This process continues iteratively until no further ancestor entries are found. This approach ensures that the retrieval process captures not only the directly relevant knowledge but also the contextual and hierarchical relationships that are critical for tasks such as SQL generation or business rule interpretation.

To illustrate this process, consider the right table in Figure 4. The retrieval begins with the entry "Campus Hiring", whose `Anc. Name` field contains the ancestor entry "Campus Recruitment". The system then retrieves "Campus Recruitment", whose `Anc. Name` field includes "Recruitment Channels". Next, "Recruitment Channels" is retrieved, which further links to "Personnel Table". Finally, "Personnel Table" is retrieved, whose `Anc. Name` field points to "Company HR Database". Since "Company HR Database" is not present in the knowledge base, the retrieval process terminates. This example demonstrates how the multi-hop traversal strategy effectively captures a chain of related knowledge entries, ensuring comprehensive retrieval of contextually relevant information.

The coarse retrieval phase typically yields a large set of candidate knowledge entries. To refine these results, we perform a fine retrieval phase, which focuses on improving precision by reranking and filtering the candidate set. We employ a state-of-the-art

re-ranking model, BGE [9], to rank the candidate nodes based on their relevance to the query. For tasks such as SQL generation, we further refine the results by leveraging a LLM to filter out irrelevant columns. The LLM evaluates the semantic relevance of each column to the user query and removes those that do not contribute to the intended SQL query. This step ensures that the final set of retrieved knowledge is both concise and highly relevant to the task.

By combining coarse retrieval, relationship-aware multi-hop traversal, re-ranking, and LLM-based filtering, SiriusBI achieves a robust and efficient knowledge retrieval process. This approach not only ensures high recall and precision but also adapts to the complex dependencies and relationships inherent in domain-specific knowledge, making it well-suited for a wide range of BI applications.

3.3 Multi-Round Dialogue with Querying

Multi-**R**ound **D**ialogue with **Q**uerying (**MRD-Q**) is a critical component in SiriusBI that receives and process user inputs for data integrity. It is designed to support MRD through automatic semantic completion and user-involved intent querying. A detailed introduction is presented in the following sections.

3.3.1 Dialogue Analysis. The multi-round dialogue analysis mechanism in SiriusBI is built upon the basic dialogue analysis module in previous work[46]. It first determines the semantic integrity of the user query by assessing whether the current input or historical conversations contain *metrics* and *dimensions*. In the context of business intelligence, dimensions refer to attributes that provide granularity level information of the user query, such as time periods or geographical locations. For example, "last quarter" is a time dimension and "in New York" is of location dimension. Metrics, on the other hand, are quantitative measurements, such as the number of sales, which might be represented by a column like `Total_Sales`. When a query includes both dimensions and metrics, such as "the number of sales in New York last month", it is possible to construct a corresponding SQL query. Conversely, if a query lacks either a dimension or a metric, such as "How about Los Angeles?", it becomes challenging to generate a meaningful SQL query. Although this approach is somewhat intuitive, it effectively addresses most dialogue patterns encountered in BI scenarios.

If both are found, the current input is considered of semantic integrity, and the workflow proceeds to the next module. If either is missing, the system supplements the current input's semantics by selecting the most recent historical input that includes metrics and dimensions, and then moves to the next step.

Unlike previous work, SiriusBI does not fine-tune a PLM (Pre-trained Language Model, e.g., [46] adopts ERNIE [67]) to determine the semantic integrity or to select relevant historical conversations. PLMs, such as BERT [15], GPT-2 [61], and ERNIE [67], are large-scale neural networks pre-trained on vast amounts of text data to capture general language representations. While they have shown remarkable performance in various natural language processing tasks, fine-tuning them for specific applications often demands substantial labeled data and computational resources. Additionally, fine-tuned models may not generalize well across different domains, limiting their transferability and effectiveness in diverse BI scenarios. Instead, we leverage the robust reasoning capability of LLMs to figure out the existence of "metric" and "dimension" and the historical conversations.

3.3.2 Intention Querying. For multi-round dialogues in real-world scenarios, relying solely on the dialogue history is often insufficient to complete the missing information due to the inherent complexity. To address these challenges, we enhance the basic dialogue analysis module by introducing an intent querying feature, resulting in a robust MRD-Q component. The intention querying process consists of the following steps:

Intent Classification. We use the same LLM for dialogue analysis in Section 3.3.1 and further categorize the expanded queries into three types. Denote the user input as Q, and the result of the intent classification as $I(Q)$. Formally, we classify the query into three categories, as indicated by Eqn. (1).

$$I(Q) = \begin{cases} 2, & \textit{if } Q \textit{ contains both dimensions and columns}, \\ 1, & \textit{if } Q \textit{ lacks either dimensions or columns}, \\ 0, & \textit{if } Q \textit{ is a non-BI scenario question}. \end{cases} \tag{1}$$

When $I(Q) = 0$, the query is identified as out of the BI scope. In this case, the system will politely decline the request and guide the user to ask data analysis-related questions. The dialogue analysis process will then restart. When $I(Q) = 1$, it indicates that, even after supplementing with historical inputs, the user query still lacks key information. The system will proactively ask the user to provide the missing information and then restart the analysis process. When $I(Q) = 2$, the query is complete. The system will proceed to the next step of the pipeline.

Intent Clarification. Although the queries entering this stage are semantically complete, additional processing is required to ensure the generated SQL accurately reflects the user's intent due to real-world complexities. To address this, the module first retrieves domain-specific knowledge from the knowledge base based on the user's query. For instance, the triplet associated with the term "YTD" is defined as: ("YTD", "Year To Date or Year To Day, referring to the period from the beginning of the current year to the present date.", "What is the YTD revenue of the mini-program?"). Such knowledge enhances the LLM's understanding of domain-specific terminology. Furthermore, the module retrieves task-specific materials, including prompt templates and demonstration examples, to guide the LLM in intent clarification. Specifically, the LLM either returns a semantically refined query or interactively presents users with disambiguation options to ensure input precision.

In summary, we develop the MRD-Q component, which supplements the heuristic dialogue analysis by the intent clarification capability of the LLM. Even if the user provides arbitrary answers, SiriusBI can clarify the user's true intent through guided follow-up questions, therefore improving the successful rate of the task execution.

3.4 SQL Generation

NL2SQL is the core task within the entire BI system, and the performance bottleneck continues to pose challenges in the industry. In this subsection, we will focus on the implementation approach of the SQL Generation module in SiriusBI, including table selection, as well as the SQL generation methods.

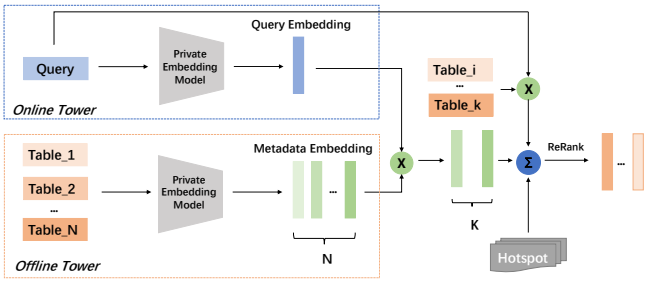


Figure 6: Overview of the table selection process.

3.4.1 Table Selection. Before generating SQL queries, it is necessary to select the appropriate tables from numerous options in an industrial database based on the user's input. This process relies on retrieving the relevant table information mentioned in the user's query. The implementation of table selection is divided into two parts: coarse ranking and re-ranking. During the coarse-ranking stage, the number of candidate tables is typically reduced by a factor of ten or more to maintain fewer than 100 candidates, thereby ensuring computational efficiency in the subsequent fine-ranking phase. In the re-ranking stage, these candidates are re-ranked to identify the most relevant ones for SQL generation.

The coarse ranking process consists of two parts: online and offline. In the offline part, each data table is subjected to a private embedding model and transformed into an embedding vector. We use the StarRocks (SR) [74] vector database to store the embeddings of tables to ensure data security and the embeddings are extracted using the general-purpose pre-trained model m3e-large [10], ensuring high-quality feature representations and domain adaptability. SR supports millisecond-level latency retrieval and allows for filtering by specified tags, making it a robust solution for efficient table selection. In the online part, we extract multiple keywords from the query through a LLM and then perform individual retrieval using each keyword. Compared to directly embedding the user's input, this approach eliminates redundant information, reduces interference from irrelevant information, and increases the recall rate of target tables.

During re-ranking, we use a scoring function balancing multiple metrics for each table t based on the hundreds of tables recalled in the coarse ranking stage. The overall calculation formula is as follows:

$$\text{Score}(t) = \text{Sim}(t) + \alpha \cdot \text{Embed}(t) + \beta \cdot \text{Heat}(t), \tag{2}$$

where $\text{Embed}(t)$ represents the semantic recall score during the coarse ranking, which is the cosine similarity score of the vectorized results. $\text{Heat}(t)$ represents the table's popularity score, which is calculated by Tencent Data Warehouse based on query logs and execution task history. $\text{Sim}(t)$ represents the similarity score between the user query and the table calculated from the field dimension, and α, β are hyperparameters.

Regarding the token-level scores, user queries frequently reference specific keywords, while tables may contain thousands of fields. Simple vectorizing the entire table's information may obscure some information details, leading to suboptimal recall performance. To address this shortcoming, we design a token-level similarity calculation to achieve better granular information recall. Denote the i-th table as t_i, and the j-th column in t_i as $c_{i,j}$. Assume that t_i contains N fields, and the input query contains K keywords. Then, the corresponding similarity score of t_i is the sum of similarity calculations between the table and K keywords. The calculation formula is as follows,

$$\text{Sim}(t_i) = \sum_{l=1}^{K} \text{sim}(k_l, t_i), \tag{3}$$

where $\text{sim}(k_l, t_i)$ represents the similarity score between keyword k_l and table t_i, which is the maximum value of the text similarity with all columns:

$$\text{sim}(k_l, t_i) = \max_{j \in \{1,2,\ldots,N\}} \text{sim}(k_l, c_{i,j}). \tag{4}$$

While TF-IDF [31] remains effective for real-time matching on short text queries, we perform coarse filtering by restricting candidate tables to those with (table count × field count) $\leq k$ dynamically adjusted per query complexity.

3.4.2 Dynamic Strategy Switching Driven by Data Conditions. In practical BI applications, the data conditions of different clients vary significantly: some clients have abundant labeled data, while others can only provide a limited number of labeled samples due to frequent business changes or data privacy restrictions. Similarly, existing SQL generation methods are often tailored to specific scenarios or domains, making migration between private scenarios costly in terms of both deployment effort and data labeling. To address this challenge, we propose a *data-condition-driven dynamic strategy switching mechanism*. This mechanism adaptively selects either a one-stage (high-precision fine-tuning) or two-stage (semantic inter- mediate representation) generation strategy based on two key factors: the volume of labeled data and the semantic similarity between domains. By doing so, it effectively balances accuracy and cost-efficiency.

Formally, we define the number of available labeled Query-SQL pairs as the Labeled Data Volume($N_{labeled}$). We also quantify the semantic similarity between the target domain and an existing labeled domain (referred to as the source domain) as Domain Similarity(S_{domain})This similarity is computed via the cosine similarity between the embeddings of schema keywords provided by the business side, as follows:

$$\text{Embed}(\text{target}) = \frac{1}{|\phi_{\text{target}}|} \sum_{k \in \phi_{\text{target}}} \text{Embed}(k),$$

$$\text{Embed}(\text{source}) = \frac{1}{|\phi_{\text{source}}|} \sum_{k \in \phi_{\text{source}}} \text{Embed}(k),$$

$$S_{\text{domain}} = \text{cosine}\Big(\text{Embed}(\text{target}), \text{Embed}(\text{source})\Big), \tag{5}$$

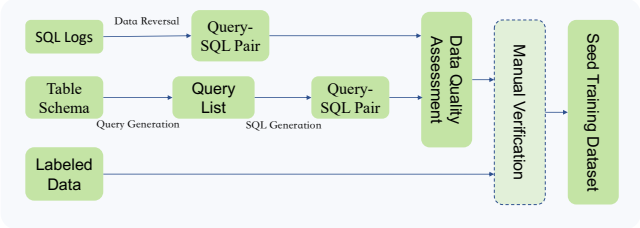
where ϕ_{target} and ϕ_{source} represent the sets of keywords for the target domain and the source domain, respectively (e.g., in the financial domain: "revenue", "quarterly report"). $\text{Embed}(\cdot)$ is Sentence-BERT [63] encoding function.

Based on the aforementioned features, we use rule-based criteria to determine the SQL generation strategy. Specifically:

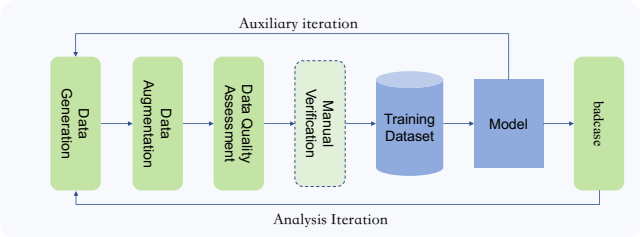
$$\text{Strategy} = \begin{cases} \text{One-Step}, & \text{if } N_{\text{labeled}} \geq \alpha \ \wedge \ S_{\text{domain}} \geq \beta \\ \text{Two-Step}, & \text{if } N_{\text{labeled}} < \alpha \ \vee \ S_{\text{domain}} < \beta \end{cases} \tag{6}$$

where α and β are hyperparameters that are typically specified based on the actual situation. In our practical experience, α is usually set to 500 and β is set to 0.7.

3.4.3 One-Step SQL Generation. To improve the performance of supervised fine-tuning and reduce annotation efforts, we have developed an automated and iterative data preparation workflow guided by manually labeled data and identified bad cases. The overall pipeline is illustrated in Figure 7, consisting of two main modules: data generation and data augmentation.



(a) Data Generation



(b) Automate Data Preparation Pipeline

Figure 7: (a) illustrates the workflow for data generation, while (b) presents the overall data preparation pipeline. In this pipeline, manual verification is an optional module.

Data Generation. To perform data generation, we need to utilize the SQL logs from the relevant scenarios and the corresponding schema information of the databases. SQL logs are the actual SQL code executed by users, representing the most authentic application data. However, since these are directly coded by users in the compiler, we need to use data reverse engineering to capture the users' true intentions. The data reverse engineering functionality is primarily implemented through LLM-based prompts. The input information includes SQL, the corresponding schema information, and proprietary domain knowledge base information. On the other hand, we generate some question sets using the schema combined with LLM, and then construct some query-SQL pairs through a general SQL generation model. The query-SQL pairs obtained through these two methods will enter the data quality evaluation module. This module mainly uses the locally deployed Qwen2.5-72B [85] model for filtering, combined with manual evaluation to select the final high-quality training set. The manually annotated data will serve as the In-Context Learning [16] (ICL) demonstrations used in the aforementioned process. The specific data generation pipeline can be seen in Figure 7a.

Data augmentation. This stage enhances data quality through two key approaches: diversity expansion and error-based negative example injection. First, we use Self-Instruct [79] with prompts to generate a broader variety of SQL queries from seed data, significantly increasing the dataset's size and diversity. These queries are then evaluated by a LLM [88], and only high-quality examples are selected for training. Second, we introduce negative examples by randomly rewriting cases from the enhanced dataset, based on error categories provided by the client. From experience, negative examples typically account for 5% of the training set, helping improve the model's robustness by teaching it to recognize and avoid common mistakes. Together, these strategies ensure a diverse, challenging dataset that enhances the model's accuracy and reliability.

Once the data generation and augmentation phases are complete, the refined dataset is used to train the model. If necessary, the pipeline can be iterated to further refine the dataset based on additional error analysis or user feedback, ensuring continuous improvement in data quality. This iterative refinement continues until the dataset meets the desired standards or until the project reaches the budget constraints specified by our clients.

Comparison with the previous work on data augmentation for NL2SQL systems. Early efforts in this field [82, 84] were generally simple, often relying on basic templates covering only a single table, heuristic methods such as synonym replacement, or traditional deep learning models like RNNs to construct NL-SQL pairs. These approaches greatly limited the quality of the generated data. More recent work, such as ScienceBenchmark [90], is closer to our method, proposing an automatic data augmentation system to expand training data. The main differences are: (1) *Seed Data Source*: We use real SQL logs combined with data reverse engineering and NL-to-SQL translation to obtain Query-SQL pairs, while ScienceBenchmark relies on manually created data. (2) *Augmentation Goals*: Besides increasing data diversity, we focus on improving model robustness by injecting noise and negative examples; ScienceBenchmark mainly aims to enlarge dataset size to formalize a high-quality benchmark. (3) *Augmentation Approach*: We directly prompt LLMs to rewrite Query-SQL pairs, whereas ScienceBenchmark uses a multi-step pipeline of template extraction, population, NL question generation, and candidate selection.

3.4.4 Two-Step SQL Generation. To address the need for cross-domain generalization in real-world scenarios and the frequent occurrence of cold-start problems in business applications, we propose a novel two-stage SQL generation paradigm. This paradigm divides the SQL generation process into two stages: Semantic Intermediate Representation (SIR) and SQL Generation.

In the first stage, the construction of SIR focuses on abstracting user intent and data semantics, transforming complex natural language queries into a standardized intermediate form that is easy to handle. The rewriting process primarily relies on in-context learning [16] to dynamically reconstruct user inputs. Additionally, we introduce a Chain-of-Thought (CoT) [81] strategy to explicitly guide the model in step-by-step reasoning about the latent intent and constraints of the query, thereby enhancing the quality of the rewritten results.

Take the query *"Analyze consumption of Tencent Cloud BI in 2023"* as an example. We first extract domain-specific knowledge which includes explanations of terms (*"Cloud BI"* in this case) and top-k demonstrations exhibiting highest similarities with the embeddings

of the query as few-shot [80] references. The input information in the SIR phase includes field descriptions retrieved from the knowledge base as well as few-shot demonstrations. The output of the SIR phase is shown below:

```
{
    "Key Components":
    {
        "Product Classification": "'Tencent
            Cloud BI' corresponds to product
            Chinese name (p_name_zh),
            specifically 'Tencent Cloud
            BI-Advanced Edition'",
        "Metric":
            "consumption_income_tax_inclusive
            (consumeincome)",
        "Time Range": 2023
    },
    "Knowledge Mapping": [
        "User's 'Tencent Cloud BI' refers to
            product Chinese name (p_name_zh)",
        "Only 'Tencent Cloud BI-Advanced
            Edition' is required for filtering"
    ],
    "Query Understanding": "User requires
        total tax-inclusive consumption income
        of 'Tencent Cloud BI-Advanced Edition'
        products in 2023 grouped by product
        name and year.",
    "Rewritten Query": "Query the total
        consumption income (tax included) (
        consumeincome ) of the product whose
        product name ( p_name_zh ) is 'Tencent
        Cloud BI-Advanced Edition' in 2023."
}
```

In the second stage, we use a code-rich LLM for SQL generation. The *"Rewriting Query"* and the schema information corresponding to the user query together form the prompt. By first constructing and refining user queries before generating SQL, this two-step approach ensures high accuracy while enabling seamless migration across diverse industrial scenarios.

4 EXPERIMENTS AND RESULTS

To evaluate our framework, in this section, we conduct an in-depth investigation from two insights: 1) On general NL2SQL tasks, SiriusBI outperforms the state-of-the-art baselines on both SRD and MRD datasets; 2) The end-to-end design of SiriusBI is rational, where the individual performance of each module demonstrates significant superiority.

4.1 Experiment Setups

4.1.1 Datasets. To demonstrate the overall performance of SiriusBI and the effectiveness of its various modules, we conduct comprehensive experiments using both academic benchmarks and industrial-level datasets on multiple tasks. We have released our carefully curated academic benchmark, MRD-BIRD. Due to data privacy concerns, we provide partially anonymized versions of two industrial-level datasets. They are publicly available at https://github.com/TencentBigData-SiriusAI/SiriusBI. Detailed descriptions of the datasets used for each task are provided below:

SRD NL2SQL. To validate the effectiveness of the SQL generation module, we utilize the *BIRD* [43], a challenging large-scale database NL2SQL evaluation benchmark aimed at narrowing the gap between academic research and practical applications. It includes 95 large databases and high-quality (user question, SQL) pairs covering 37 professional fields. Besides, we collect 182 SRD queries spanning 10669 tables and involving 43 columns from 5 different business scenarios, thereby creating a SRD dataset termed as *SRD-Industry*.

MRD NL2SQL. This task better simulates real-world BI scenarios by evaluating both intent understanding and SQL generation in multi-round dialogue (MRD) settings. Due to the lack of a standard MRD NL2SQL benchmark, we expand part of the SRD dialogues from the BIRD dataset into 3–5 round dialogues. Specifically, we use an LLM to decompose complex SQL into turn-level queries, then generate corresponding user questions that reflect natural MRD conversation style—building on prior context without repetition. We also create *MRD-Industry*, a new dataset matching the database scale of SRD-Industry, with 55 query sets spanning 2–5 rounds each (160 queries over 35 columns). To increase complexity, we select 10 sets and remove certain metrics and dimensions, and select another 10 sets in which we inject confusion by replacing metrics or dimensions with ambiguous alternatives.

Knowledge Management. To evaluate the knowledge management module, we focus on two main aspects: 1) Assessing its impact on NL2SQL tasks; 2) Evaluating the performance of knowledge retrieval. In the latter evaluation, we utilize the knowledge base corresponding to the MRD-Industry dataset, which contains knowledge across six dimensions: table, column, value, term, udf, and alias, totaling 7704 pieces of knowledge.

Data Insight. For evaluation of data insights, we select 35 descriptive requests and 45 complex requests involving descriptive analysis used in industrial scenarios. The requests involve data retrieval, extreme value calculation, correlation analysis, etc. Note that, other types of requests are not included in the dataset, as it's difficult to evaluate the correctness of answers to open-ended questions.

4.1.2 Evaluation Metrics. For NL2SQL tasks on the BIRD dataset, we adhere to the standard evaluation metrics as outlined in [43], utilizing *Execution Accuracy* (EX) and *Valid Efficiency Score* (VES). For NL2SQL tasks on our internal datasets, we follow previous works [19, 46] and employ the *Useful Execution Accuracy* (UEX) metric, which considers a query correct if its intent aligns with that of the golden query. This makes UEX more suitable for business intelligence scenarios.

4.1.3 Baselines and Model Configurations. We compare the NL2SQL capability of SiriusBI with three state-of-the-art baselines as below: (1) *DIN-SQL* [58]; (2) *MAC-SQL* [75]; (3) *MRD-SQL* [46]; (4) *SiriusBI*. For the baseline models, we followed their publicly available usage instructions as closely as possible, including base models and prompt designs. For DIN-SQL and MAC-SQL, we used their original configurations and prompts without modification. We reproduce the NL2BI module of the previous work [46] as MRD-SQL. In our method and MRD-SQL, we use Qwen2.5-Coder-32B [27] as the base

model for two-step SQL generation on SRD and MRD datasets. For one-step SQL generation on the BIRD dataset (and MRD-BIRD), we use XiYanSQL-QwenCoder-32B [22]. This choice is motivated by its demonstrated state-of-the-art performance on this benchmark, which obviates the need for additional fine-tuning. For industrial datasets, we fine-tuned Qwen2.5-Coder-32B to better fit domain data. Specifically, we use four nodes equipped with a total of 32 NVIDIA A100-40GB GPUs, setting the training to 3 epochs, a learning rate of *3e-5*, a batch size of 2, and a sequence length of 8192 tokens. The fine-tuning dataset comprises over 9,000 entries collected from real-world industrial scenarios. Overall, we ensured fair comparison by aligning baseline setups with their original methods and optimizing prompts within their intended use.

4.2 End-to-End NL2SQL Performance

In this experiment, we evaluate the end-to-end NL2SQL performance of various baselines on the SRD and MRD datasets. Note that, as an analysis module following NL2SQL, the data insight module is not included in this part.

4.2.1 Results on SRD Datasets. We conduct experiments on two SRD datasets—BIRD's dev set and SRD-Industry—with results summarized in Table 1. Our method, SiriusBI, employing both one-step and two-step SQL generation, consistently outperforms baseline approaches. The improvement is especially pronounced on the SRD-Industry dataset, where SiriusBI achieves at least a 40% increase in UEX. This significant gain is attributed to SiriusBI 's explicit design to tackle real-world challenges such as ambiguous fields and wide tables, which general-purpose models often struggle with. We exclude MRD-SQL from this comparison since it differs from SiriusBI only in the multi-round dialogue analysis module, leading to identical performance on SRD datasets. Additionally, the one-step variant of SiriusBI, fine-tuned on domain-specific data, surpasses the two-step method by 4.05% in UEX. This suggests that supervised fine-tuning provides a measurable advantage in handling domain-specific queries, further enhancing its performance.

Table 1: Evaluations on the BIRD's dev set and SRD-Industry for SRD NL2SQL tasks.

Methods	BIRD		SRD-Industry
	EX(%)	VES(%)	UEX(%)
DIN-SQL + GPT-4o	50.72	58.79	30.06
MAC-SQL + GPT-4o	57.56	58.04	36.42
SiriusBI *two-step*	<u>65.32</u>	<u>67.87</u>	<u>79.19</u>
SiriusBI *one-step*	**68.97**	**70.89**	**83.24**

4.2.2 Results on MRD Datasets. Next, we evaluate SiriusBI on two MRD datasets: MRD-BIRD and MRD-Industry, and present the experimental results in Table 2. Our findings reveal that NL2SQL methods designed for single-round dialogue perform poorly on MRD datasets. Specifically, the UEX scores of DIN-SQL and MAC-SQL on the MRD-Industry dataset fall below 16%, highlighting their limitations in handling multi-round dialogue scenarios. In contrast, the two-step and one-step variants of SiriusBI achieve UEX scores of 55.00% and 62.50%, respectively, demonstrating the consistent superiority compared to MRD-SQL in real-world NL2SQL tasks.

Table 2: Evaluations on the MRD-BIRD and MRD-Industry for MRD NL2SQL tasks.

Methods	MRD-BIRD	MRD-Industry
	EX(%)	UEX(%)
DIN-SQL + GPT-4o	41.69	11.25
MAC-SQL + GPT-4o	42.35	15.63
MRD-SQL *two-step*	46.98	22.50
MRD-SQL *one-step*	<u>50.61</u>	28.13
SiriusBI *two-step*	48.03	<u>55.00</u>
SiriusBI *one-step*	**51.14**	**62.50**

It is worth noting that on the MRD-BIRD dataset, the performance gap between SiriusBI and MRD-SQL is narrower. Specifically, the one-step variant of MRD-SQL achieves a higher EX score than the two-step variant of SiriusBI. We attribute this phenomenon to the absence of user interaction for intent clarification in the experiments on MRD-BIRD, which diminishes the advantage of SiriusBI. However, by incorporating domain-specific knowledge during intent clarification, SiriusBI still outperforms MRD-SQL when utilizing the same SQL generation method, underscoring the effectiveness of our approach in leveraging domain-specific insights to enhance performance.

4.2.3 In-production Results. In addition to experiments on industrial datasets, we evaluate SiriusBI in production across multiple Tencent business domains. Before deployment, we apply the proposed data-condition-driven dynamic strategy switching mechanism to select the most suitable SQL generation approach based on each domain's data characteristics. For Tencent Finance and Advertising, clients provide sufficient labeled data ($N_{\text{labeled}} \geq \alpha$) and the source and target domains are identical, resulting in a semantic similarity $S_{\text{domain}} = 1 \geq \beta$. Under these conditions, the mechanism selects the one-step fine-tuning strategy, achieving in-production accuracies of 97% and 93%, respectively. Conversely, Tencent Cloud serves diverse users with limited labeled data and low domain similarity between our available labeled data. Thus, the mechanism chooses the two-step strategy, which attains an average accuracy of 96% across 14 randomly selected clients.

4.2.4 Discussion. We observe that the in-production results are higher than those reported on the SRD and MRD datasets. This is mainly due to three factors: 1) Knowledge Completeness: Although (MRD-)BIRD datasets provide extra knowledge, their coverage is limited compared to our comprehensive internal knowledge base, which better supports accurate SQL generation in real scenarios. 2) Data Distribution and Query Complexity: Real-world user queries in specific applications tend to be more uniform and simpler, allowing fine-tuned models to achieve higher accuracy. In contrast, SRD and MRD contain diverse and complex questions from multiple domains, making them more challenging benchmarks. 3) Iterative Feedback and Model Refinement: In-production models benefit from multiple rounds of feedback and continuous improvement guided by real failure cases (see Section 3.4.3). SRD and MRD results, however, reflect single-cycle training without such iterative updates, limiting their performance. In summary, richer knowledge, simpler real-world queries, and ongoing model refinement explain the higher accuracy observed in production compared to benchmark datasets.

4.3 Detailed Analysis

4.3.1 Evaluation of MRD-Q. In this part, we present the ablation study on dialogue analysis and intention querying using the MRD-Industry dataset. The experimental results are displayed in Table 3. Based on the experimental results, we can observe that, both the SiriusBI *one-step* and SiriusBI *two-step* heavily rely on the dialogue analysis and the intention querying modules. Specifically, when both modules are removed (without MRD-Q), the performance of SiriusBI declines significantly, with drops of 15.62% and 34.38% for one-step and two-step designs, respectively. Similarly, the absence of the intention querying module results in a performance degradation of 6.25% and 21.25% for SiriusBI. This indicates that the dialogue analysis and intention querying modules are critical to the overall effectiveness of SiriusBI.

Table 3: Ablation study of MRD-Q on the MRD dataset. "Q" refers to the Intention Querying module.

Baseline	UEX	Δ
SiriusBI *two-step w/o MRD-Q*	20.62%	-34.38%
SiriusBI *two-step w/o Q*	33.75%	-21.25%
SiriusBI *two-step*	55.00%	-
SiriusBI *one-step w/o MRD-Q*	46.88%	-15.62%
SiriusBI *one-step w/o Q*	56.25%	-6.25%
SiriusBI *one-step*	62.50%	-

In addition, the ablation study reveals that SiriusBI *one-step* fine-tuned on a private dataset exhibits greater robustness compared to SiriusBI *two-step*. Concretely, the impact of removing intention querying in SiriusBI *one-step* is less pronounced, resulting in a performance drop of only 6.25%, compared to a more significant decline of 21.25% for SiriusBI *two-step*. The difference can be attributed to the fact that SiriusBI *one-step* acquires domain-specific knowledge from the training data during the learning process, which helps to mitigate the negative effects of lacking conversational understanding and intention querying.

4.3.2 Evaluation of Table Selection. While table selection directly impacts the UEX of SQL selection, we conduct an ablation study on this component. We use the queries in the SRD dataset and evaluate the performance using the Recall@5 metric as described before. The experimental results are shown in Table 4.

Table 4: Ablation study of table selection. "Embed" and "Heat" refer to the embedding and heat information, respectively.

Baseline	Recall@5	Δ
SiriusBI *w/o Embed & Heat*	77.47%	-13.74%
SiriusBI *w/o Heat*	79.12%	-12.01%
SiriusBI *w/o Embed*	83.52%	-7.69%
SiriusBI	**91.21%**	-

It is clear that both the embedding and heat information enhance the recall performance of table selection, which indicates the effectiveness of the combined ranking function in Equation 2.

4.3.3 Evaluation of Knowledge Management. Then, we examine the impact of knowledge management (KM) on the NL2SQL pipeline. The results of disabling this module on the MRD dataset are presented in Table 5. The findings reveal a significant decline in performance for SiriusBI when KM is turned off. Concretely, UEX decreases by 23.12% for SiriusBI *one-step* and 26.87% for SiriusBI *two-step*. These results highlight the critical role of the knowledge management module in enhancing the overall performance of NL2SQL, as both intention querying and SQL generation heavily depend on the availability of relevant knowledge.

Next, we evaluate the module using a dataset for knowledge retrieval with 7704 samples. In this evaluation, we compare SiriusBI against two widely used baseline models: BM25 [65] and Vector [32], both of which are prominent in the field of text retrieval. We assess the performance of these methods using two key metrics: KR Recall measuring the recall rate for retrieving relevant knowledge entries across all keywords and SL Recall measuring the recall rate specifically for linking table columns. The experimental results demonstrate the following performance: BM25 achieves a KR Recall of 90.28% and an SL Recall of 89.58%; Vector achieves a KR Recall of 93.06% and an SL Recall of 93.06%; SiriusBI achieves a KR Recall of 95.14% and an SL Recall of 93.75%. SiriusBI outperforms the baselines across both metrics, thereby ensuring that other modules leveraging knowledge management within SiriusBI can access accurate and relevant additional information.

Table 5: Ablation study of knowledge management on MRD Dataset. "KM" refers to Knowledge Management.

Baseline	UEX	Δ
SiriusBI *two-step w/o KM*	28.13%	-26.87%
SiriusBI *two-step*	55.00%	-
SiriusBI *one-step w/o KM*	39.38%	-23.12%
SiriusBI *one-step*	62.50%	-

4.3.4 Data Insight Evaluations. Finally, we conduct experiments on the Data Insight module. Remind that the data insight dataset contains 80 quantifiable questions involving comprehensive analysis and complex task planning.

For evaluation, we use real-world queries from the financial domain and conduct comparative experiments with three powerful LLM baselines: GPT-4o [72], Qwen2.5 [27] and Commnd R+ [14]. The accuracy of responses to these queries serves as the evaluation criterion, and to ensure the reliability of the experimental results, each query is tested three times.

The results demonstrate that the Data Insight module of SiriusBI achieves an accuracy of 95%, significantly outperforming the baseline models. Specifically, Command-R+ achieves an accuracy of 75%, Qwen2.5 achieves 80%, and GPT-4o achieves 83%. These findings indicate that the Data Insight module of SiriusBI maintains a notable performance advantage even against the most effective LLMs currently in use. This superior performance can be attributed to the function call architecture in SiriusBI, which enables it to outperform GPT-4o by 12% in comprehensive data analysis tasks.

Specifically, for complex task planning, we compare the Data Insight module with a method that only performs tool invocation on the aforementioned high-quality queries. Compared to the simple tool invocation method, the MRD-agent architecture adopted by

SiriusBI shows a significant advantage, improving the accuracy from 70% to 100% on this dataset.

4.4 Case Study

In this part, we use an exemplar to demonstrate the whole workflow through a BI interaction. At the beginning, the user raises a question – *What is the income of the Company A in 2024?* The Dialogue Analysis module evaluates the semantic integrity of this query. In fact, this query contains both a "dimension" (2024) and a "metric" (income), making it semantically complete. The query then proceeds to the second stage.

The Intention Querying module recalls relevant information from the knowledge base based on the input query. Specifically, a piece of retrieved knowledge is a triplet – (company, by default, without special instructions, refers to Tencent, "What is gross profit of the company in 2023?"), which provides an explanation for the term "company". The Intention Querying module further prompts the user for clarification as there still exists ambiguity within the query. In this case, the word "*income*" is ambiguous (with tax or not), the user will be asked whether it refers to the column "*shouldincome*" or "*shouldincome_after*".

Afterwards, the SQL generation module selects the appropriate tables and translates the user query into an SQL query. The generated SQL statement is `SELECT SUM(shouldincome_after) AS total_income FROM revenue_by_quarter WHERE YEAR (ftime) = 2024 AND cname = Company A`.

The Data Insight module receives the SQL queries and their corresponding execution results. Based on the user's additional insight requirements like *"Perform dimension attribution analysis on the revenue of Company A in 2024"*, the module analyzes the user's intent, decomposes tasks through the Planner, performs data preparation and invokes attribution tools, and finally generates an insight report, as shown in Figure 8.

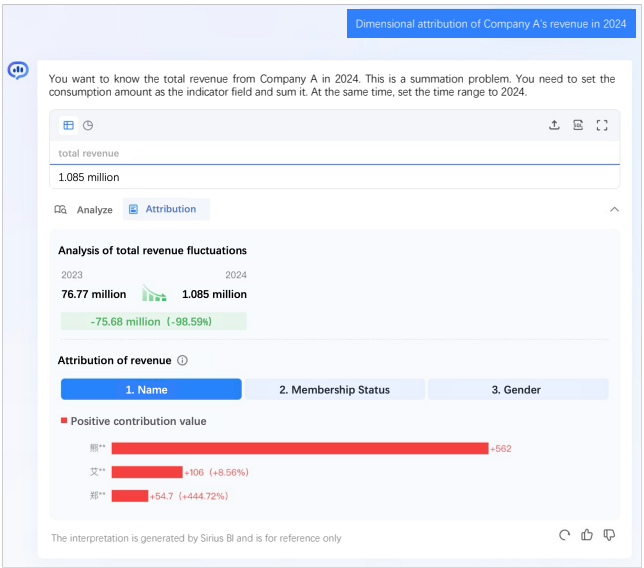


Figure 8: Showcase of practical application of dimensional attribution in data insights.

4.5 Utility and Usability Study

To evaluate whether SiriusBI effectively improves users' productivity and experience in real-world scenarios, we conducted a gray-release test within a business analysis team at Tencent data platform. This study aimed to quantify the system's impact on operational efficiency and gather subjective feedback on usability. We randomly divided 40 business analysts into two groups – Group A (20 analysts): granted access to SiriusBI for daily tasks; Group B (20 analysts): continued using legacy tools. The experiment lasted for two weeks, during which we monitored both groups' workflows. To ensure comparability, we controlled that tasks were distributed evenly across groups based on complexity and historical completion times. We find that Group A exhibited a 30.2% reduction in average task completion time compared to Group B. This efficiency gain is attributed to SiriusBI's integrated workflow and advanced technical design. What's more, Group A provided an average SUS score of 84.2/100 (SD = 6.5), categorizing SiriusBI as "excellent" according to the SUS benchmark [4]. Key strengths highlighted in feedback included: (1) 89% of users reported that domain-specific knowledge grounding reduced manual schema lookup time. (2) 92% agreed that SiriusBI's data insight module "provided reliable support for business decision-making" (e.g., identifying root causes of revenue fluctuations). The gray-release test confirms that SiriusBI enhances both productivity and user experience by addressing the functionality deficiencies.

5 CONCLUSION

In this paper, we propose SiriusBI, a LLM-powered solution for business intelligence, specifically designed to address three key challenges encountered in industrial business intelligence scenarios. Our system 1) establishes a comprehensive end-to-end workflow of the entire business intelligence process to address functionality deficiencies; 2) implements multi-round dialogue with querying capabilities to overcome interaction limitations; 3) adopts a data-conditioned SQL generation method selection strategy to alleviate the great cost brought by cross-domain migration in SQL generation. Currently, SiriusBI is deployed across Tencent's finance, advertising, and cloud sectors, serving enterprise clients with various profiles. Extensive experiments on public benchmark and industrial datasets demonstrate that SiriusBI clearly outperforms competitive LLM-based baselines in both SRD and MRD-based NL2SQL tasks. Detailed analysis on different modules demonstrate the rationality of our design. In addition, ablation studies show that each component of SiriusBI contributes significantly to the overall performance, indicating its robustness and scalability across domains. User studies further confirm that SiriusBI enhances both productivity and user experience.

ACKNOWLEDGMENTS

This work is supported by National Natural Science Foundation of China (92470121, 62402016), National Key R&D Program of China (2024YFA1014003), and High-performance Computing Platform of Peking University. We also thank the engineers from the Data Platform department, Technology and Engineering Group (TEG) of Tencent for their technical and engineering support. The co-correspondence authors are Yang Li and Wentao Zhang.

REFERENCES

[1] Katrin Affolter, Kurt Stockinger, and Abraham Bernstein. 2019. A comparative survey of recent natural language interfaces for databases. *The VLDB Journal* (2019).

[2] Ranjita Bhagwan, Rahul Kumar, Ramachandran Ramjee, George Varghese, Surjyakanta Mohapatra, Hemanth Manoharan, and Piyush Shah. 2014. Adtributor: Revenue debugging in advertising systems. In *11th USENIX Symposium on Networked Systems Design and Implementation (NSDI 14)*.

[3] George EP Box, Gwilym M Jenkins, Gregory C Reinsel, and Greta M Ljung. 2015. *Time series analysis: forecasting and control*. John Wiley & Sons.

[4] John Brooke et al. 1996. SUS-A quick and dirty usability scale. *Usability evaluation in industry* (1996).

[5] Hasan Alp Caferoğlu and Özgür Ulusoy. 2024. E-SQL: Direct Schema Linking via Question Enrichment in Text-to-SQL. *arXiv preprint arXiv:2409.16751* (2024).

[6] Ruichu Cai, Jinjie Yuan, Boyan Xu, and Zhifeng Hao. 2021. Sadga: Structure-aware dual graph aggregation network for text-to-sql. *Advances in Neural Information Processing Systems* 34 (2021), 7664–7676.

[7] Ruisheng Cao, Lu Chen, Zhi Chen, Yanbin Zhao, Su Zhu, and Kai Yu. 2021. LGESQL: Line Graph Enhanced Text-to-SQL Model with Mixed Local and Non-Local Relations. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing*. 2541–2555.

[8] Surajit Chaudhuri, Umeshwar Dayal, and Vivek Narasayya. 2011. An overview of business intelligence technology. *Commun. ACM* (2011).

[9] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216* (2024).

[10] Jianlv Chen, Shitao Xiao, Peitian Zhang, Kun Luo, Defu Lian, and Zheng Liu. 2024. Bge m3-embedding: Multi-lingual, multi-functionality, multi-granularity text embeddings through self-knowledge distillation. *arXiv preprint arXiv:2402.03216* (2024).

[11] Xiaoxin Chen, Meng Wu, and Mangning Wang. 2024. Application of business intelligence under deep neural network in credit scoring of bank users. *Journal of Computational Methods in Sciences and Engineering* (2024).

[12] Hilary Cheng, Yi-Chuan Lu, and Calvin Sheu. 2009. An ontology-based business intelligence application in a financial knowledge management system. *Expert Systems with Applications* (2009).

[13] DongHyun Choi, Myeong Cheol Shin, EungGyun Kim, and Dong Ryeol Shin. 2021. Ryansql: Recursively applying sketch-based slot fillings for complex text-to-sql in cross-domain databases. *Computational Linguistics* 47, 2 (2021), 309–332.

[14] cohere.ai Team. 2024. The Cohere Platform. https://docs.cohere.com

[15] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*.

[16] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Jingyuan Ma, Rui Li, Heming Xia, Jingjing Xu, Zhiyong Wu, Tianyu Liu, et al. 2022. A survey on in-context learning. *arXiv preprint arXiv:2301.00234* (2022).

[17] Xuemei Dong, Chao Zhang, Yuhang Ge, Yuren Mao, Yunjun Gao, Jinshu Lin, Dongfang Lou, et al. 2023. C3: Zero-shot text-to-sql with chatgpt. *arXiv preprint arXiv:2307.07306* (2023).

[18] BV Elasticsearch. 2018. Elasticsearch. *software], version* 6, 1 (2018).

[19] Avrilia Floratou, Fotis Psallidas, Fuheng Zhao, Shaleen Deep, Gunther Hagleither, Wangda Tan, Joyce Cahoon, Rana Alotaibi, Jordan Henkel, Abhik Singla, et al. 2024. NL2SQL is a solved problem... Not!. In *CIDR*.

[20] Yujian Gan, Xinyun Chen, Jinxia Xie, Matthew Purver, John R Woodward, John Drake, and Qiaofu Zhang. 2021. Natural SQL: Making SQL Easier to Infer from Natural Language Specifications. *arXiv preprint arXiv:2109.05153* (2021).

[21] Min Gao, Zheng Li, Ruichen Li, Chenhao Cui, Xinyuan Chen, Bodian Ye, Yupeng Li, Weiwei Gu, Qingyuan Gong, Xin Wang, et al. 2023. EasyGraph: A multi-functional, cross-platform, and effective library for interdisciplinary network analysis. *Patterns* 4, 10 (2023), 100839.

[22] Yingqi Gao, Yifu Liu, Xiaoxia Li, Xiaorong Shi, Yin Zhu, Yiming Wang, Shiqi Li, Wei Li, Yuntao Hong, Zhiling Luo, Jinyang Gao, Liyu Mou, and Yu Li. 2024. XiYan-SQL: A Multi-Generator Ensemble Framework for Text-to-SQL. *arXiv preprint arXiv:2411.08599* (2024).

[23] Louie Giray. 2023. Prompt engineering with ChatGPT: a guide for academic writers. *Annals of biomedical engineering* 51, 12 (2023), 2629–2633.

[24] Marco Hegger and Adriana Saraceni. 2024. Smart Port Sustainability: A Business Intelligence Framework for CO2 Reduction in Cargo Truck Operations. In *Advances in Production Management Systems. Production Management Systems for Volatile, Uncertain, Complex, and Ambiguous Environments - 43rd IFIP WG 5.7 International Conference*, Vol. 728. 309–323.

[25] Vagelis Hristidis, Luis Gravano, and Yannis Papakonstantinou. 2003. Efficient IR-Style Keyword Search over Relational Databases. In *Proceedings of 29th International Conference on Very Large Data Bases*. 850–861.

[26] Vagelis Hristidis and Yannis Papakonstantinou. 2002. DISCOVER: Keyword Search in Relational Databases. In *Proceedings of 28th International Conference on Very Large Data Bases*. 670–681.

[27] Binyuan Hui, Jian Yang, Zeyu Cui, Jiaxi Yang, Dayiheng Liu, Lei Zhang, Tianyu Liu, Jiajun Zhang, Bowen Yu, Kai Dang, et al. 2024. Qwen2. 5-Coder Technical Report. *arXiv preprint arXiv:2409.12186* (2024).

[28] Kyle Hundman, Valentino Constantinou, Christopher Laporte, Ian Colwell, and Tom Soderstrom. 2018. Detecting spacecraft anomalies using lstms and nonparametric dynamic thresholding. In *Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining*.

[29] Ashwin Ittoo, Antal van den Bosch, et al. 2016. Text analytics in industry: Challenges, desiderata and trends. *Computers in Industry* 78 (2016), 96–107.

[30] Srinivasan Iyer, Ioannis Konstas, Alvin Cheung, Jayant Krishnamurthy, and Luke Zettlemoyer. 2017. Learning a Neural Semantic Parser from User Feedback. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*. 963–973.

[31] Thorsten Joachims. 1997. A Probabilistic Analysis of the Rocchio Algorithm with TFIDF for Text Categorization. In *Proceedings of the Fourteenth International Conference on Machine Learning*. 143–151.

[32] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick S. H. Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. 2020. Dense Passage Retrieval for Open-Domain Question Answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing*.

[33] Jagbir Kaur, Ashok Choppadandi, Pradeep Kumar Chenchala, Varun Nakra, and Pandi Kirupa Gopalakrishna Pandian. 2019. Case Studies on Improving User Interaction and Satisfaction using AI-Enabled Chatbots for Customer Service. *International Journal of Transcontinental Discoveries* 6, 1 (2019), 29–34.

[34] Wenjun Ke, Ziyu Shang, Zhizhao Luo, Peng Wang, Yikai Guo, Qi Liu, and Yuxuan Chen. 2024. Unveiling factuality and injecting knowledge for LLMs via reinforcement learning and data proportion. *Science China Information Sciences* (2024).

[35] Sehoon Kim, Suhong Moon, Ryan Tabrizi, Nicholas Lee, Michael W Mahoney, Kurt Keutzer, and Amir Gholami. 2024. An LLM Compiler for Parallel Function Calling. In *Forty-first International Conference on Machine Learning*.

[36] David Pidsley Julian Sun Georgia O'Callaghan Christopher Long Kevin Quinn Fay Fei Edgar Macari Jamie O'Brien Kurt Schlegel, Anirudh Ganeshan. 2023. Magic Quadrant for Analytics and Business Intelligence Platforms. https://www.gartner.com/doc/reprints?id=1-2HW1JC8Q&ct=240620

[37] Chee Sun Lee, Peck Yeng Sharon Cheang, and Massoud Moslehpour. 2022. Predictive analytics in business analytics: decision tree. *Advances in Decision Sciences* 26, 1 (2022), 1–29.

[38] Boyan Li, Yuyu Luo, Chengliang Chai, Guoliang Li, and Nan Tang. 2024. The Dawn of Natural Language to SQL: Are We Fully Ready? [Experiment, Analysis \u0026 Benchmark]. *Proceedings of the Very Large Data Bases Endowment* 17, 11 (2024), 3318–3331.

[39] Chaofan Li, Yingxia Shao, and Zheng Liu. 2024. SEA-SQL: Semantic-Enhanced Text-to-SQL with Adaptive Refinement. *arXiv preprint arXiv:2408.04919* (2024).

[40] Fei Li and H. V. Jagadish. 2014. Constructing an Interactive Natural Language Interface for Relational Databases. *Proceedings of the Very Large Data Bases Endowment* 8, 1 (2014), 73–84.

[41] Guoliang Li, Jiang Wang, and Guo Chen. 2024. openGauss: An Enterprise-Grade Open-Source Database System. *J. Comput. Sci. Technol.* (2024).

[42] Haoyang Li, Jing Zhang, Hanbing Liu, Ju Fan, Xiaokang Zhang, Jun Zhu, Renjie Wei, Hongyan Pan, Cuiping Li, and Hong Chen. 2024. CodeS: Towards Building Open-source Language Models for Text-to-SQL. *Proceedings of the ACM on Management of Data* 2, 3 (2024), 1–28.

[43] Jinyang Li, Binyuan Hui, Ge Qu, Jiaxi Yang, Binhua Li, Bowen Li, Bailin Wang, Bowen Qin, Ruiying Geng, Nan Huo, Xuanhe Zhou, Chenhao Ma, Guoliang Li, Kevin Chen-Chuan Chang, Fei Huang, Reynold Cheng, and Yongbin Li. 2023. Can LLM Already Serve as A Database Interface? A BIg Bench for Large-Scale Database Grounded Text-to-SQLs. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023*.

[44] Jian-Zhong Li. 2024. opengauss: An open-source database for the era of artificial intelligence. *Journal of Computer Science and Technology* (2024).

[45] Zeyan Li, Chengyang Luo, Yiwei Zhao, Yongqian Sun, Kaixin Sui, Xiping Wang, Dapeng Liu, Xing Jin, Qi Wang, and Dan Pei. 2019. Generic and robust localization of multi-dimensional root causes. In *2019 IEEE 30th International Symposium on Software Reliability Engineering (ISSRE)*.

[46] Jinqing Lian, Xinyi Liu, Yingxia Shao, Yang Dong, Ming Wang, Zhang Wei, Tianqi Wan, Ming Dong, and Hailin Yan. 2024. ChatBI: Towards Natural Language to Complex Business Intelligence SQL. *arXiv preprint arXiv:2405.00527* (2024).

[47] Ahmed Ali Linkon, Mujiba Shaima, Md Shohail Uddin Sarker, Norun Nabi, Md Nasir Uddin Rana, Sandip Kumar Ghosh, Mohammad Anisur Rahman, Hammed Esa, Faiaz Rahat Chowdhury, et al. 2024. Advancements and applications of generative artificial intelligence and large language models on business management: A comprehensive review. *Journal of Computer Science and Technology Studies* 6, 1 (2024), 225–232.

[48] Xinyu Liu, Shuyu Shen, Boyan Li, Peixian Ma, Runzhi Jiang, Yuyu Luo, Yuxin Zhang, Ju Fan, Guoliang Li, and Nan Tang. 2024. A Survey of NL2SQL with Large Language Models: Where are we, and where are we going? *arXiv preprint arXiv:2408.05109* (2024).

[49] Scott M Lundberg and Su-In Lee. 2017. A unified approach to interpreting model predictions. *Advances in neural information processing systems* 30 (2017).

[50] Yi Luo, Xuemin Lin, Wei Wang, and Xiaofang Zhou. 2007. Spark: top-k keyword query in relational databases. In *Proceedings of the International Conference on Management of Data*. ACM, 115–126.

[51] Karime Maamari, Fadhil Abubaker, Daniel Jaroslawicz, and Amine Mhedhbi. 2024. The Death of Schema Linking? Text-to-SQL in the Age of Well-Reasoned Language Models. In *NeurIPS 2024 Third Table Representation Learning Workshop*.

[52] Venkata Vamsikrishna Meduri, Abdul Quamar, Chuan Lei, Vasilis Efthymiou, and Fatma Ozcan. 2021. BI-REC: guided data analysis for conversational business intelligence. *arXiv preprint arXiv:2105.00467* (2021).

[53] Sharan Narang and Aakanksha Chowdhery. 2022. Pathways language model (palm): Scaling to 540 billion parameters for breakthrough performance. *Google AI Blog* (2022).

[54] Solomon Negash. 2004. Business Intelligence. *Communications of the Association for Information Systems* 177, 195 (2004), 177.

[55] Mohammed T Nuseir. 2021. Designing business intelligence (BI) for production, distribution and customer services: a case study of a UAE-based organization. *Business Process Management Journal* (2021).

[56] Ana-Maria Popescu, Alex Armanasu, Oren Etzioni, David Ko, and Alexander Yates. 2004. Modern Natural Language Interfaces to Databases: Composing Statistical Parsing with Semantic Tractability. In *Proceedings of the 20th international conference on Computational Linguistics*.

[57] Mohammadreza Pourreza, Hailong Li, Ruoxi Sun, Yeounoh Chung, Shayan Talaei, Gaurav Tarlok Kakkar, Yu Gan, Amin Saberi, Fatma Ozcan, and Sercan O Arik. 2024. CHASE-SQL: Multi-Path Reasoning and Preference Optimized Candidate Selection in Text-to-SQL. *arXiv preprint arXiv:2410.01943* (2024).

[58] Mohammadreza Pourreza and Davood Rafiei. 2023. DIN-SQL: Decomposed In-Context Learning of Text-to-SQL with Self-Correction. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023*.

[59] Mohammadreza Pourreza and Davood Rafiei. 2024. DTS-SQL: Decomposed Text-to-SQL with Small Large Language Models. *arXiv preprint arXiv:2402.01117* (2024).

[60] Xiaoru Qu, Yifan Wang, Zhao Li, and Jun Gao. 2024. Graph-Enhanced Prompt Learning for Personalized Review Generation. *Data Sci. Eng.* (2024).

[61] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, et al. 2019. Language models are unsupervised multitask learners. *OpenAI blog* (2019).

[62] Nitin Rane, Mallikarjuna Paramesha, Saurabh Choudhary, and Jayesh Rane. 2024. Business Intelligence and Business Analytics With Artificial Intelligence and Machine Learning: Trends, Techniques, and Opportunities. *Techniques, and Opportunities* (2024).

[63] Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*.

[64] Kevin Mejía Rivera. 2024. Developing a Media Watcher Through Business Intelligence Tools: An Approach for Emerging Media Industries. In *Gaming, Entertainment, and Media Conference*. 1–4.

[65] SE Robertson and K Sparck Jones. 1976. Relevance Weighting of Search Terms. *Journal of the American Society for Information Science* (1976).

[66] Ruoxi Sun, Sercan Ö Arik, Alex Muzio, Lesly Miculicich, Satya Gundabathula, Pengcheng Yin, Hanjun Dai, Hootan Nakhost, Rajarishi Sinha, Zifeng Wang, et al. 2023. SQL-PaLM: Improved Large Language Model Adaptation for Text-to-SQL (extended). *arXiv preprint arXiv:2306.00739* (2023).

[67] Yu Sun, Shuohuan Wang, Yukun Li, Shikun Feng, Xuyi Chen, Han Zhang, Xin Tian, Danxiang Zhu, Hao Tian, and Hua Wu. 2019. Ernie: Enhanced representation through knowledge integration. *arXiv preprint arXiv:1904.09223* (2019).

[68] Shayan Talaei, Mohammadreza Pourreza, Yu-Chen Chang, Azalia Mirhoseini, and Amin Saberi. 2024. CHESS: Contextual Harnessing for Efficient SQL Synthesis. *arXiv preprint arXiv:2405.16755* (2024).

[69] Minsang D Tamang, Vinod Kumar Shukla, Shaista Anwar, and Ritu Punhani. 2021. Improving business intelligence through machine learning algorithms. In *2021 2nd International Conference on Intelligent Engineering and Management*. 63–68.

[70] Claude Team. 2024. Claude3. https://anthropic.com/claude

[71] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, et al. 2023. Gemini: a family of highly capable multimodal models. *arXiv preprint arXiv:2312.11805* (2023).

[72] OpenAI team. 2022. Introduction of OpenAI Text Generation APIs. https://platform.openai.com/docs/guides/text-generation

[73] OpenAI team. 2023. OpenAI CodeX. https://openai.com/index/openai-codex

[74] StarRocks team. 2023. StarRocks. https://github.com/StarRocks/starrocks

[75] Bing Wang, Changyu Ren, Jian Yang, Xinnian Liang, Jiaqi Bai, Qian-Wen Zhang, Zhao Yan, and Zhoujun Li. 2023. MAC-SQL: A Multi-Agent Collaborative Framework for Text-to-SQL. *arXiv preprint arXiv:2312.11242* (2023).

[76] Bailin Wang, Richard Shin, Xiaodong Liu, Oleksandr Polozov, and Matthew Richardson. 2020. RAT-SQL: Relation-Aware Schema Encoding and Linking for Text-to-SQL Parsers. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*. 7567–7578.

[77] Chenglong Wang, Alvin Cheung, and Rastislav Bodík. 2017. Synthesizing highly expressive SQL queries from input-output examples. In *Proceedings of the 38th Conference on Programming Language Design and Implementation*. 452–466.

[78] Liang Wang, Nan Yang, Xiaolong Huang, Linjun Yang, Rangan Majumder, and Furu Wei. 2024. Multilingual e5 text embeddings: A technical report. *arXiv preprint arXiv:2402.05672* (2024).

[79] Yizhong Wang, Yeganeh Kordi, Swaroop Mishra, Alisa Liu, Noah A Smith, Daniel Khashabi, and Hannaneh Hajishirzi. 2023. Self-Instruct: Aligning Language Models with Self-Generated Instructions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*.

[80] Yaqing Wang, Quanming Yao, James T Kwok, and Lionel M Ni. 2020. Generalizing from a few examples: A survey on few-shot learning. *ACM computing surveys (csur)* (2020).

[81] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed H Chi, Quoc V Le, and Denny Zhou. 2022. Chain-of-thought prompting elicits reasoning in large language models. In *Proceedings of the 36th International Conference on Neural Information Processing Systems*. 24824–24837.

[82] Nathaniel Weir, Prasetya Utama, Alex Galakatos, Andrew Crotty, Amir Ilkhechi, Shekar Ramaswamy, Rohin Bhushan, Nadja Geisler, Benjamin Hättasch, Steffen Eger, et al. 2020. Dbpal: A fully pluggable nl2sql training pipeline. In *Proceedings of the 2020 ACM SIGMOD International Conference on Management of Data*.

[83] Bernhard Wieder and Maria-Luise Ossimitz. 2015. The impact of Business Intelligence on the quality of decision making–a mediation model. *Procedia Computer Science* 64 (2015), 1163–1171.

[84] Kun Wu, Lijie Wang, Zhenghua Li, Ao Zhang, Xinyan Xiao, Hua Wu, Min Zhang, and Haifeng Wang. 2021. Data augmentation with hierarchical SQL-to-question generation for cross-domain text-to-SQL parsing. *arXiv preprint arXiv:2103.02227* (2021).

[85] An Yang, Baosong Yang, Beichen Zhang, Binyuan Hui, Bo Zheng, Bowen Yu, Chengyuan Li, Dayiheng Liu, Fei Huang, Haoran Wei, et al. 2024. Qwen2. 5 technical report. *arXiv preprint arXiv:2412.15115* (2024).

[86] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. 2023. React: Synergizing reasoning and acting in language models. In *International Conference on Learning Representations (ICLR)*.

[87] Zhong Zeng, Mong-Li Lee, and Tok Wang Ling. 2016. Answering Keyword Queries involving Aggregates and GROUPBY on Relational Databases. In *Proceedings of the 19th International Conference on Extending Database Technology*. 161–172.

[88] Baoli Zhang, Haining Xie, Pengfan Du, Junhao Chen, Pengfei Cao, Yubo Chen, Shengping Liu, Kang Liu, and Jun Zhao. 2023. ZhuJiu: A Multi-dimensional, Multi-faceted Chinese Benchmark for Large Language Models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*.

[89] Jiayu Zhang, Jiangrong Shen, Zeke Wang, Qinghai Guo, Rui Yan, Gang Pan, and Huajin Tang. 2024. SpikingMiniLM: energy-efficient spiking transformer for natural language understanding. *Science China Information Sciences* (2024).

[90] Yi Zhang, Jan Deriu, George Katsogiannis-Meimarakis, Catherine Kosten, Georgia Koutrika, and Kurt Stockinger. 2023. Sciencebenchmark: A complex real-world benchmark for evaluating natural language to sql systems. *arXiv preprint arXiv:2306.04743* (2023).

[91] Xuanhe Zhou, Zhaoyan Sun, and Guoliang Li. 2024. Db-gpt: Large language model meets database. *Data Science and Engineering* (2024).