Thesis for the Degree of Master of Science in Big Data Analytics

An LLM-based Question Answering System over Relational Databases via Text-to-SQL

Sol Jeong

Department of Big Data Analytics
Graduate School
Kyung Hee University
Yongin, Korea

February, 2026

An LLM-based Question Answering System over Relational Databases via Text-to-SQL

Sol Jeong

Department of Big Data Analytics
Graduate School
Kyung Hee University
Yongin, Korea

February, 2026

An LLM-based Question Answering System over Relational Databases via Text-to-SQL

by

Sol Jeong

Advised by

Dr. Jae-Yoon Jung

Submitted to the Department of Big Data Analytics
and the Faculty of the Graduate School of
Kyung Hee University in partial fulfillment
of the requirements for degree of
Master of Science in Big Data Analytics

Dissertation Committee :

Chairman Shin Jungwoo

Younghoon Kim

Jae-Yoon Jung

Table of Contents

List of Tables

List of Figures

Abstract

An LLM-based Question Answering System over Relational Databases via Text-to-SQL

by Sol Jeong
Master of Science in Big Data Analytics
Graduate School of Kyung Hee University
Advised by Dr. Jae-Yoon Jung

As digital transformation accelerates in manufacturing, vast amounts of industrial Internet of Things (IoT) sensor data are being accumulated. However, enabling shop-floor workers without Structured Query Language (SQL) expertise to query this data using natural language remains a significant challenge. Although natural language interfaces powered by Large Language Models (LLMs) have recently gained attention, conventional Text-to-SQL approaches for manufacturing IoT data are hindered by wide-table schemas that expand the search space and induce hallucinations, as well as by high-frequency time-series data that require costly aggregation, thereby limiting real-time responsiveness. To address these challenges, this study proposes Star-TAG, an LLM-based question answering framework that integrates star-schema–based IoT data mart with a Table-Augmented Generation (TAG) pipeline. The framework restructures wide-table IoT data into a star schema with time-based aggregation and employs schema-aware prompting that explicitly models fact–dimension relationships for the LLM. Experiments on real-world battery manufacturing data show that Star-TAG achieves 72.0% execution accuracy on synthetic benchmarks, a 44.0 percentage-point improvement over wide-table baselines, while reducing token usage by 51.1%. In addition, lightweight open-source models attain performance comparable to commercial models under Star-TAG, indicating that schema-driven optimization can mitigate model-scale limitations. These results demonstrate that efficient LLM-based natural language interfaces can substantially enhance data accessibility for shop-floor workers in manufacturing environments subject to security and computational constraints.

Key words

Text-to-SQL; Table-Augmented Generation (TAG); Large Language Model (LLM); Data Mart; Industrial Internet of Things (IIoT); Relational Database

1 Introduction

Modern manufacturing is undergoing accelerated digital transformation through the adoption of smart factories, leading to the accumulation of vast amounts of industrial Internet of Things (IoT) data on shop floors (Tao et al., 2019; Zhong et al., 2017). Although Human-Machine Interfaces (HMIs) have been widely used to deliver operational information to workers in such environments, it remains difficult for shop-floor personnel to directly access and interpret insights from this massive volume of data (Kumar & Lee, 2022). This is because conventional HMI dashboards cannot effectively address workers' specific queries about complex shop floor data (Mourtzis et al., 2022), and it is impractical to predefine and distribute all potentially relevant data through fixed mobile or HMI interfaces.

Recently, large language models (LLMs) have enabled natural language–based question answering (QA) systems for shop-floor applications via mobile devices and industrial applications (Zhang et al., 2025). However, most existing studies focus on unstructured data such as work instructions or equipment manuals (Keman Freire et al., 2024; Garcia et al., 2024), or on general relational database (RDB) processing. In contrast, QA systems designed specifically for high-frequency sensor-based IoT data—which are central to real-world manufacturing operations—remain largely unexplored.

Although Text-to-SQL techniques have been extensively studied for natural language access to RDB, directly applying them to manufacturing IoT data introduces significant challenges. First, sensor measurements are often stored in heterogeneous and schema-inefficient formats, most notably in wide-table schemas in which tens to hundreds of similarly named sensor attributes are arranged horizontally in a single table. Such designs substantially increase the contextual complexity that LLMs must process and expand the search space for column selection, thereby degrading Structured Query Language (SQL) generation accuracy (Li et al., 2023; Lei et al., 2024). As a result, LLMs may select incorrect columns or generate non-existent attributes, leading to hallucinations (Cao et al., 2024; Qu et al., 2024).

Second, manufacturing IoT systems generate massive volumes of time-series data at sub-second granularity. Although sensors may record values at millisecond or second intervals (Wang et al., 2023), shop-floor workers typically require aggregated information over meaningful time windows, such as "the average temperature of equipment X at 3 PM." Under

high-frequency storage schemes, even a simple one-hour query may involve scanning and aggregating thousands of records, resulting in high latency and computational overhead. Consequently, conventional Text-to-SQL approaches struggle to deliver responsive and reliable QA for industrial IoT data, limiting their practical applicability on the shop floor.

To overcome these challenges, this study proposes an LLM-based QA framework that can reliably answer questions about industrial IoT data on shop floors. While existing Text-to-SQL research has focused on improving models or prompting strategies given a fixed schema, this approach adopts a data-centric methodology that proactively designs DB schemas optimized for LLM reasoning (Zha et al., 2025). Specifically, sensor DBs stored in RDBs are restructured into star schema-based data marts suitable for analysis (Kimball & Ross, 2013), and schema-aware prompting strategies that are easier for LLMs to interpret are employed. As a result, Table-Augmented Generation (TAG)—an end-to-end pipeline from natural language queries to SQL generation and natural language responses—is implemented (Biswal et al., 2024), enabling shop-floor workers without SQL expertise to query industrial data and obtain analytical responses in natural language.

To empirically validate the effectiveness of the proposed system, a QA system was built using one week of operational data from battery manufacturing equipment. After transforming the raw IoT data from the RDB into a star schema-based data mart, the QA system was implemented using both a commercial LLM accessed via API (GPT-4.1) and open-source LLMs deployable on-premise (Gemma3-12B and Qwen3-4B). For evaluation, experiments were conducted using a real-world shop-floor use-case dataset and a synthetic dataset generated following the OmniSQL methodology (Li et al., 2025). The results indicate that, across all three models, the proposed system achieved an average 44.0 percentage-point improvement in execution accuracy over wide-table baseline while reducing token usage by 51.1%. Furthermore, the success rate after self-correction reached 100% in the star-schema environment, demonstrating that schema design improves not only accuracy but also system robustness.

The main contributions of this study are as follows:

- **Data-centric Text-to-SQL approach:** Unlike existing research that focuses on model and prompt engineering, this study presents a new direction for enhancing LLM performance through schema-driven optimization. The impact of star schema-based data marts on LLM

SQL generation accuracy, token efficiency, and error recovery is empirically demonstrated.

- **Application of the TAG pipeline to manufacturing IoT domain:** An end-to-end QA system encompassing natural language answer generation beyond SQL generation is applied to manufacturing IoT data, enabling shop-floor workers to perform data-driven decision making without SQL expertise.
- **Empirical validation and practical guidelines:** The effectiveness of the methodology is demonstrated through both use-case-based evaluation on real Manufacturing Execution System (MES) data and quantitative benchmarking using OmniSQL-based synthetic datasets. Additionally, common error types observed in the experiments are analyzed, and actionable guidelines for real-world deployment are provided.

The remainder of this paper is organized as follows. Chapter 2 reviews related work on LLM-based industrial assistants and Text-to-SQL. Chapter 3 presents the system architecture, star schema-based data mart design strategy, and TAG pipeline including Schema-Aware Prompting. Chapter 4 describes the experimental setup, reports performance comparisons, and presents the implemented system interfaces. Chapter 5 analyzes Text-to-SQL error patterns with self-correction mechanisms and discusses the structural factors underlying Star-TAG's performance improvements. Finally, Chapter 6 presents conclusions and future research directions.

2 Related Work

2.1 Digital Intelligent Assistance based on Large Language Model

Digital Intelligent Assistants (DIAs) are intelligent software applications that provide services through natural language interaction with users (Maedche et al., 2019; Wellsandt et al., 2022). DIAs consist of a combination of conversational agents (McTear, 2022) and technology-based agents, which interpret user intent and achieve task objectives by integrating with external systems (Wellsandt et al., 2022).

The technical components of DIAs are defined as five core elements (Wellsandt et al., 2022): Speech-to-Text (STT) for voice input recognition, natural language understanding for intent parsing, dialog management for conversation flow control, external information system connections for data access, and Text-to-Speech (TTS) for response delivery. Among these, connectivity to external information systems is a key factor that determines the practical capability and intelligence level of DIAs (Wellsandt et al., 2022). In manufacturing environments, data-driven decision making on the shop floor becomes possible only when DIAs can access production DBs, equipment status, and operational metrics.

Traditional DIA research in industrial settings has primarily focused on predictive maintenance and worker support. Wellsandt et al. (2022) proposed a five-stage predictive maintenance framework—Sense-Detect-Predict-Decide-Act—where DIAs served as interfaces to communicate analysis results from rule-based systems to maintenance personnel. However, this approach relied on predefined rules and rigid workflows, limiting its ability to flexibly respond to diverse natural language queries.

Recent advances in LLMs have enabled more flexible DIA implementations in manufacturing environments. Yuan et al. (2025) proposed “Chat with MES,” demonstrating that MESs can be operated through natural language interfaces. Colabianchi et al. (2024) evaluated the applicability of LLM-based DIAs in assembly-line manufacturing environments. These studies showed that LLMs can interpret ambiguous user inputs and decompose complex tasks into multi-step actions.

However, existing LLM-based DIA research has primarily addressed system control interfaces or unstructured document retrieval, and natural language QA over IoT sensor data stored in RDB remains underexplored. Shop-floor operators and managers require insights into

equipment performance, product quality, and production trends without SQL expertise. To achieve this, DIAs must be able to translate users' natural language questions into executable DB queries and return accurate analytical results. This study addresses this gap by implementing the DIA's external information system connection as Text-to-SQL-based natural language data querying mechanism.

2.2 Table-Augmented Generation via Text-to-SQL

Recently, Retrieval-Augmented Generation (RAG) technology has been widely utilized for QA on unstructured documents (Lewis et al., 2020). However, since RAG relies on vector similarity-based document retrieval, it cannot accurately perform relational operations such as aggregation, filtering, and joins on structured data stored in RDBs. Therefore, accurate TAG requires a mechanism that interprets users' intent from natural language and generates SQL queries.

Technology for querying structured data in RDBs through natural language has been studied for a long time. The core technology, Text-to-SQL, is a task that converts users' natural language questions into executable SQL queries (Hong et al., 2025). This technology originated from rule-based systems in the 1970s, evolved through LSTM and Transformer-based deep learning approaches in the 2010s, and advanced to methods leveraging pre-trained language models (PLMs) such as BERT and RoBERTa. Recently, the emergence of LLMs has established a new paradigm in the Text-to-SQL field.

Key techniques for LLM-based Text-to-SQL include In-context Learning (ICL) (Brown et al., 2020) and Chain-of-Thought (CoT) (Wei et al., 2022) prompting. ICL guides the model to learn SQL generation patterns by including a few examples in the prompt, while CoT enhances reasoning accuracy by decomposing complex query generation processes into explicit steps. The performance of these techniques is evaluated through the Spider (Yu et al., 2018) and BIRD (Li et al., 2023) benchmarks. Spider is a cross-domain benchmark containing over 200 DBs, while BIRD raises practical difficulty by including questions that require real DB environments and external knowledge.

However, Text-to-SQL only addresses the stage of converting natural language questions to SQL and does not cover the end-to-end QA pipeline that returns results to users in natural

language. To address this limitation, Biswal et al. (2024) proposed TAG. TAG is a framework that integrates the entire process of natural language-based data querying, consisting of three stages: (1) Query Synthesis – converting natural language questions to SQL queries (corresponding to Text-to-SQL), (2) Query Execution – executing queries to extract data, and (3) Answer Generation – producing natural language responses based on extracted data. In other words, while Text-to-SQL was limited to returning tables or scalar values, TAG delivers user-friendly natural-language answers, enabling shop-floor workers without SQL knowledge to gain insights from data.

Since this study targets structured IoT sensor data, it adopts the Text-to-SQL-based TAG approach. When applying the TAG paradigm to manufacturing IoT DBs, the inherent schema complexity becomes problematic, hindering LLM schema comprehension and inducing hallucinations. To address this, this study ensures TAG reliability by building a TAG pipeline that applies Text-to-SQL after reducing schema complexity via star schema-based data mart design (Kimball & Ross, 2013), which has been widely utilized for analytical query optimization.

3 Framework

3.1 Requirements

This chapter introduces a framework for natural language QA on manufacturing IoT sensor data and defines three key design requirements that the system must satisfy. These requirements are derived from the limitations of existing Text-to-SQL approaches and guide the subsequent data mart and TAG pipeline design.

Easy query for non-experts. The system must enable shop-floor workers without SQL expertise to query data using natural language and obtain analytical answers. Existing Text-to-SQL systems often only return query execution results as tables or scalar values. However, since shop-floor workers are not accustomed to interpreting raw SQL outputs, context-aware natural language responses are required rather than simple numerical listings. To address this, chapter 3.4 introduces a TAG pipeline with an Interpretive Answer Generation technique that converts SQL execution results into contextualized natural language responses.

Accurate query results and relevant answer. The QA system must generate correct SQL queries for users' natural language questions and return accurate execution results. When applying existing Text-to-SQL approaches to IoT data, hallucinations occur because LLMs fail to identify correct attributes or generate non-existent columns in wide-table schemas, where tens to hundreds of attributes are listed in a single table. Therefore, to reduce the LLM search space and improve schema comprehension, chapter 3.3 proposes a star schema-based data mart design.

Low token usage and fast response time. The QA system must minimize prompt token consumption and query processing latency. IoT sensors in manufacturing processes collect data at sub-second resolution, resulting in massive time-series datasets. When such high-frequency data are queried directly, even simple aggregate queries require scanning thousands of records, and large numbers of tokens are needed to encode wide-table schemas in prompts. This leads to increased response latency and API costs. To address this, chapter 3.3 introduces time-based pre-aggregation (Temporal Aggregation) and compact schema representations to improve efficiency.

3.2 System Overview

Star-TAG is a framework that integrates a star schema-based data mart with a TAG

pipeline. As shown in Figure 1, the proposed framework consists of two phases: the build-time phase, in which the IoT data mart is constructed, and the run-time phase, in which TAG is executed via Text-to-SQL.

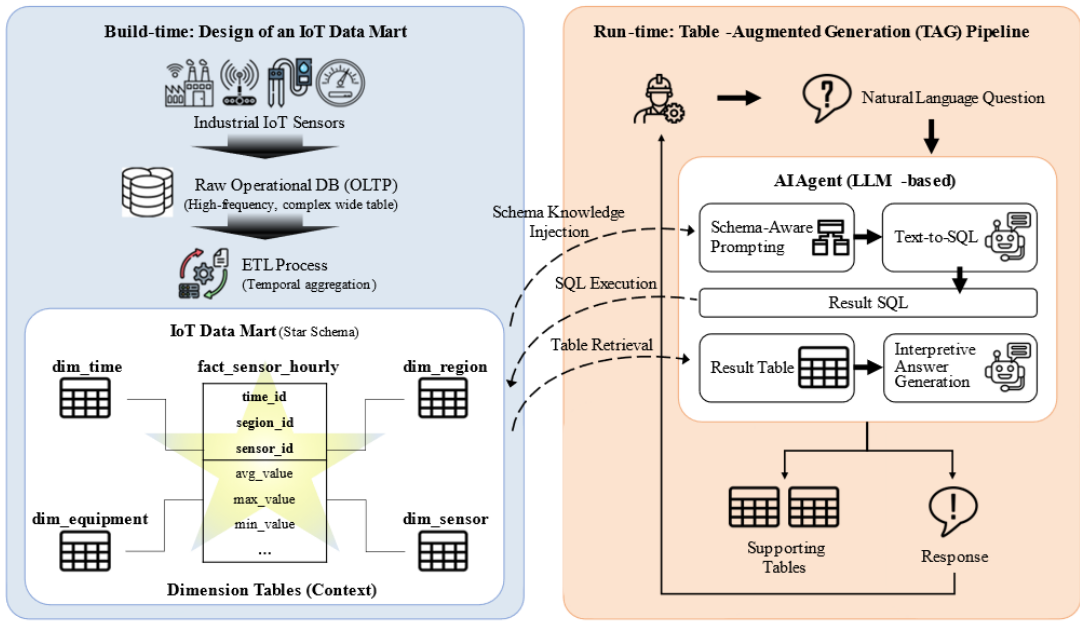


Figure 1. The Star-TAG framework for industrial IoT data analysis via data mart–augmented generation.

Build-time: Design of an IoT Data Mart. In this phase, a data mart tailored for natural language QA is constructed from the original operational DB. IoT sensor data from manufacturing shop floors are generated at high frequency by Online Transaction Processing (OLTP) systems and are typically stored in RDBs as wide tables with complex semantics. The Extract, Transform, and Load (ETL) process applies time-based aggregation (Temporal Aggregation) to this raw data, transforming it into analysis-ready representation. The transformed data are loaded into a data mart with a star-schema structure, consisting of a central Fact table (sensor measurements) surrounded by Dimension tables that provide contextual information such as time, location, equipment, and sensor attributes.

Run-time: Table-Augmented Generation (TAG) Pipeline. In this phase, users' natural language queries are processed based on the prestructured data mart. The two phases are

connected through Schema Knowledge Injection, whereby the data mart's schema and semantic information are injected into the run-time prompt. When shop-floor workers submit questions in natural language, the LLM-based Agent generates responses through the TAG pipeline—Query Synthesis, Query Execution, and Interpretive Answer Generation—as illustrated in the right-hand panel of Figure 1.

3.3 Design of an IoT Data Mart

The MES DB on shop floors is an OLTP system for real-time transaction processing, and performing analytical queries directly on the operational DB is inappropriate from the perspectives of system load and stability. The analytical QA targeted by this study requires Online Analytical Processing (OLAP) characteristics such as aggregation, filtering, and multidimensional analysis. To meet these requirements, a dedicated data mart optimized for OLAP workloads is designed.

When utilizing the original MES DB directly for Text-to-SQL, two problems arise. First, massive volumes of rows accumulate due to high-frequency, sub-second sensor data. Second, the DB has a wide-table schema, in which all sensor measurements are stored as columns in a single table. These characteristics significantly expand the LLM's search space and hinder schema comprehension, thereby degrading SQL generation accuracy. To address these problems, this study restructures the data repository by transforming the wide-table schema into a star schema.

3.3.1 Star Schema for IoTs

To store IoT data in RDBs, the wide-table schema, which integrates multiple sensor attributes into a single table, is commonly used. A wide-table schema is a data model that stores each sensor measurement as a separate column and is widely used in sensor-cloud and IoT platforms (Ma & Yang, 2014; Blondheim, 2025). In industrial IoT environments, hundreds of sensors installed on a single piece of equipment generate data concurrently, making it common to store area-specific sensor measurements as separate columns (Wang et al., 2023). For example, in a battery dry room, temperature, differential pressure, and supply-fan speed for each region are stored as individual columns. The original data in this study also follows this

structure, with region, sensor type, and measurement semantics encoded in column names (e.g., REGION_01_TEMPERATURE_PV, REGION_01_DIFF_PRESSURE_UPPER, REGION_01_SUPPLY_FAN_RPM). In such a flat schema, the LLM must identify columns relevant to the question among numerous similarly named attributes, which can lead to hallucinations in which incorrect columns are selected.

The star schema is a standard data warehouse modeling technique consisting of a central Fact table surrounded by Dimension tables. The Fact table stores measurements that are the subject of analysis, while Dimension tables define the context in which those measurements occurred. In this study, hourly aggregated sensor statistics are modeled as Facts, and four analytical axes constituting the measurement context are modeled as Dimensions (see Figure 2). The time dimension (dim_time) specifies time attributes of measurement points, the equipment dimension (dim_equipment) specifies equipment information, the region dimension (dim_region) specifies specific location information within equipment, and the sensor dimension (dim_sensor) describes the sensor types deployed at each location. Raw sensor readings are aggregated into hourly maxima, minima, and averages and stored in the Fact table.

Figure 2. Schema transformation from a wide-table format to a star schema for industrial IoT sensor data.

The choice of a star schema is motivated by its compatibility with LLM-based SQL generation. First, hierarchical dimension designs such as Snowflake Schema increase JOIN depth and complicate query synthesis for LLMs. In contrast, the star schema requires only single-hop joins between the Fact and Dimension tables, resulting in simpler and more predictable SQL structures. Second, LLMs have been exposed to OLAP-style query patterns—filtering by Dimensions and aggregating from Facts—during pre-training, making the fact–dimension separation of the star schema naturally aligned with their learned representations.

This star-schema design improves Text-to-SQL performance in two ways. First, schema complexity is significantly reduced. As shown in Figure 2, a wide table with 69 columns used in this study is restructured into four Dimension tables and one Fact table, substantially simplifying the schema information that the LLM must process in the prompt. Second, the explicit separation of roles between Fact and Dimension tables clarifies SQL generation patterns. In analytical queries, Dimension tables supply filtering conditions (the SQL WHERE clause) and grouping criteria (the SQL GROUP BY clause), while the Fact table becomes the target of aggregate functions (e.g., MAX, MIN, AVG, and SUM). This clear role separation guides the LLM to select the correct JOIN path and aggregate syntax. Moreover, the star schema offers superior extensibility: when new sensors or equipment are introduced, wide tables require costly schema alterations (ALTER TABLE), whereas the star schema accommodates such changes by adding rows to the relevant Dimension tables, thereby preserving schema stability.

3.3.2 Temporal Aggregation of IoT Data

To address the data-volume challenge, time-based aggregation (Temporal Aggregation) is applied during the Extract, Transform, and Load (ETL) process. The original data records sensor readings at sub-second resolution, yielding approximately 148,000 records over seven days. These data are aggregated into one-hour windows and transformed into summary statistics, including mean, maximum, minimum, and null ratio for each interval. Through this process, the dataset is reduced to approximately 3,700 records, achieving nearly a 40-fold reduction in physical search space.

The rationale for adopting a one-hour aggregation interval is as follows. The queries that

shop-floor workers and managers typically require are aggregated values over specific time points or intervals, such as “the average temperature of equipment X at 3 PM″ or “the time period when the highest pressure occurred yesterday.” Directly querying raw sub-second data is only necessary in specialized scenarios, such as anomaly detection or fine-grained diagnostics. The one-hour resolution aligns with the temporal granularity commonly used on the shop floor, including shift-based analysis, daily trend monitoring, and time-window comparisons. This pre-aggregation enables the LLM to operate on meaningful statistical units rather than individual measurements, allowing analytical queries to be processed with simple SELECT statements. For example, the question “What was the average temperature of equipment X last week?” would otherwise require on-the-fly aggregation over hundreds of thousands of records, whereas in the data mart it can be answered by directly querying precomputed hourly averages. This simplifies SQL generation and reduces error likelihood by eliminating unnecessary aggregation operations.

3.4 Table-Augmented Generation Pipeline

The TAG pipeline answers users’ natural language queries based on the preconstructed data mart. The build-time phase and run-time phase are connected through Schema Knowledge Injection, whereby schema metadata from the data mart constructed during build-time—including table definitions and foreign key relationships—is extracted in Data Definition Language (DDL) format and injected into the LLM prompt at run time, enabling the LLM to comprehend the data mart structure and generate accurate SQL queries.

When shop-floor workers submit questions in natural language, the LLM-based agent produces responses through the three-stage pipeline illustrated in Figure 1:

Query Synthesis. The user’s natural language question is translated into an executable SQL query. This stage consists of two substeps. First, in Schema-Aware Prompting, a prompt is constructed by combining the DDL-based schema description with the user’s question. Then, in Text-to-SQL generation, the constructed prompt is fed to the LLM to produce an SQL query.

Query Execution. The generated SQL query is executed against the IoT data mart, and the resulting table is returned. The star-schema design ensures efficient joins between Fact and Dimension tables, enabling low-latency query processing.

Interpretive Answer Generation. The returned result table is transformed into a natural language answer. In this stage, the user's original question and the query results are provided to the LLM, which generates a contextualized natural language response that includes interpretation of the results along with supporting tables for reference.

In the Interpretive Answer Generation stage, results obtained through query execution are converted into user-friendly natural language answers. Existing Text-to-SQL systems often only return query execution results in tabular form. However, since shop-floor workers are not accustomed to interpreting raw SQL outputs, context-aware responses are required rather than simple numerical listings. Accordingly, this study proposes an Interpretive Answer Generation module to produce responses that include both interpretation and contextual information about query results.

Interpretive Answer Generation is designed to generate responses containing four elements. First, a direct answer to the question is provided with appropriate units (e.g., "The average temperature of REGION_01 is 96.3℃"). Second, contextual interpretation of the result is added (e.g., "This is within the normal operating range"). Third, when multiple values are returned, trend analysis is performed (e.g., "There is a gradually increasing trend from August 1st to 4th"). Fourth, anomalies or actionable alerts are presented when applicable. Queries on the shop floor are systematically categorized into types such as single-value lookup, period-based trends, inter-zone comparisons, statistical verification, and threshold-based filtering. Accordingly, an adaptive few-shot strategy is applied that dynamically selects response examples based on the detected question type. Specifically, the question type is identified via keyword-based classification, and only type-relevant examples are included in the prompt. This approach reduces prompt token usage while eliciting responses that are better aligned with the characteristics of each question type.

Prompt Template 1. Prompt for Interpretive Answer Generation

```
You are an assistant for shop-floor workers.

## Question
{question}

## SQL Result
{result_table}

## Response Guidelines
1. Direct answer: State values with units
2. Interpretation: Normal/high/low, trend, or comparison
3. Keep concise (2-3 sentences), no technical jargon

## Example for few-shot strategy
{selected_example}

Response:
```

4 Implementation and Experiments

4.1 Experimental Setup

Experiments were conducted using three LLMs: GPT-4.1 (accessed via a commercial API from OpenAI), Gemma3-12B (a Google-developed open-source model with 12 billion parameters), and Qwen3-4B (an open-source model with 4 billion parameters developed by Alibaba). The inclusion of lightweight open-source models, often referred to as small language models (SLMs), alongside a commercial LLM was motivated by deployment constraints commonly observed in manufacturing environments, where strict security policies, air-gapped networks, and data governance requirements often prohibit outbound API calls. Under such conditions, on-premise deployable SLMs represent a practical and scalable alternative for industrial applications.

The open-source models were executed on a server equipped with two NVIDIA TITAN RTX GPUs, providing a total of 48 GB of VRAM. For data storage, PostgreSQL 16.3, an open-source relational database management system, was employed. To ensure reproducibility and fair comparison, identical generation parameters were applied across all models (temperature = 0; max_tokens = 4,096; seed = 42).

The primary objective of this experimental study was to evaluate the impact of schema design on Text-to-SQL performance, specifically comparing wide-table and star schema-based data mart configurations. To this end, two complementary evaluation datasets were constructed.

Query set A: Industry requirements-based queries. Twenty questions were constructed based on a User Requirements Specification (URS) provided by an industrial partner. The URS reflects realistic information needs of shop-floor operators and engineers in a manufacturing environment. These questions were categorized into four types: simple lookup (5), conditional filtering (5), aggregation and statistics (5), and complex join and comparison (5). Because identical questions were posed to both schemas, Query set A enables controlled comparison of schema impact while holding query content constant.

Query set B: Synthetic benchmark queries. This dataset was generated using the data synthesis methodology of OmniSQL (Li et al., 2025). OmniSQL generates query data at four difficulty levels: Simple, Moderate, Complex, and Highly Complex. The Simple level was selected because typical analytical queries in manufacturing environments—such as sensor

lookups and threshold-based filtering—primarily involve SELECT operations with straightforward WHERE conditions. Since OmniSQL generates question–SQL pairs conditioned on the input schema, 94 questions were independently generated for each schema. This design allows Query set B to serve as a schema-aware benchmark that complements the requirements-based evaluation of Query set A.

Text-to-SQL performance was evaluated using four metrics, each capturing a distinct aspect of system quality:

Execution Success Rate (ESR): The proportion of generated SQL queries that execute successfully on the database without syntax or execution errors. ESR measures the LLM's ability to produce syntactically valid and executable SQL, serving as a baseline indicator of query generation reliability.

Execution Accuracy (EX): A Text-to-SQL evaluation metric proposed in the BIRD benchmark (Li et al., 2023). EX determines whether the generated SQL and the ground-truth SQL return identical result sets when executed on the same database. Unlike the traditional Exact Match (EM) approach, which requires exact string matching of SQL queries, EX evaluates semantic equivalence at the execution level, allowing structurally different but logically equivalent SQL queries to be counted as correct.

Token Usage (TU): The total number of tokens consumed during SQL generation (input + output). TU reflects the efficiency of schema representation and SQL generation, directly impacting API costs for commercial models and context window utilization for resource-constrained deployments.

Response Time (RT): The elapsed time from natural language question input to SQL generation completion. RT assesses the system's suitability for interactive shop-floor applications, where near-instantaneous feedback is required to support effective decision making by shop-floor operators.

4.2 Experimental Results

To isolate the effect of schema design, the proposed Star-TAG framework was compared against a wide-table baseline. Both configurations utilized the same industrial IoT data repository and were evaluated using an identical TAG pipeline, differing only in the underlying data schema design. Table 1 summarizes the performance comparison across all models and query sets.

Query set A, constructed based on the URS, was applied to all three LLMs for comparison (Table 1, Query set A). Across all models, Star-TAG consistently outperformed the wide-table baseline in both execution reliability and semantic accuracy. In terms of Execution success rate (ESR), Star-TAG achieved 90.0–95.0%, compared to 55.0–95.0% for the wide table, with improvements of up to 40.0 percentage points observed for Gemma3-12B. For Execution Accuracy (EX), the improvements were more pronounced: Star-TAG achieved 55.0–70.0%, whereas the wide table achieved only 5.0–30.0%, representing gains of 35.0 to 65.0 percentage points across the three models. Token usage decreased by 21.9–36.7%, demonstrating the efficiency of the compact star schema representation. Response time exhibited model-dependent behavior, with no consistent trend observed across configurations.

For Query set B, the 94 question–SQL pairs generated for each schema were applied to all three models (Table 1, Query set B). Star-TAG achieved Execution Accuracy values ranging from 66.0% to 75.5%, compared to 18.1–38.3% for the wide table, representing gains of 36.2 to 57.4 percentage points across the three models. Execution success rate also improved consistently, with Star-TAG achieving 97.9–100.0%, compared to 88.3–97.9% for the wide table. A key finding emerges from the substantial gap between ESR and EX in the wide-table condition. Despite achieving relatively high ESR (88.3–97.9%), wide table's EX remained low (18.1–38.3%). This disparity indicates that a substantial proportion of generated queries executed successfully but returned incorrect results, suggesting that LLMs struggle to reliably select the correct columns among numerous similarly named attributes in wide-table schemas, a phenomenon commonly referred to as column selection hallucination.

Table 1. Performance comparison of Text-to-SQL results

Dataset	Metric[1)]	GPT-4.1 (1,800B)[2)]			Gemma3 (12B)			Qwen3 (4B)		
		Wide-table	Star-TAG	Improve.	Wide-table	Star-TAG	Improve.	Wide-table	Star-TAG	Improve.
Query set A	ESR [%]	**95.00**[3)]	**95.00**	-	55.00	**95.00**	40.00	65.00	90.00	25.00
	EX [%]	30.00	65.00	35.00	5.00	**70.00**	65.00	5.00	55.00	50.00
	Avg. TU [#]	1,832	**1,160**	672	1,792	1,201	591	1,879	1,467	412
	Avg. RT [s]	6.20	**5.31**	0.89	9.56	9.61	-0.05	11.35	12.75	-1.4
Query set B	ESR [%]	97.90	**100.00**	2.10	88.30	97.90	9.60	88.30	97.90	9.60
	EX [%]	38.30	74.50	36.20	27.70	66.00	38.30	18.10	**75.50**	57.40
	Avg. TU [#]	1,860	848	1,012	1,649	**772**	877	1,784	969	815
	Avg. RT [s]	6.47	6.85	- 0.38	5.66	**3.18**	2.48	9.90	6.93	2.97

[1)] *ESR: Execution success rate, EX: Execution accuracy, TU: Token usage, RT: Response time*

[2)] *The number of parameters is not publicly disclosed; the value is estimated.*

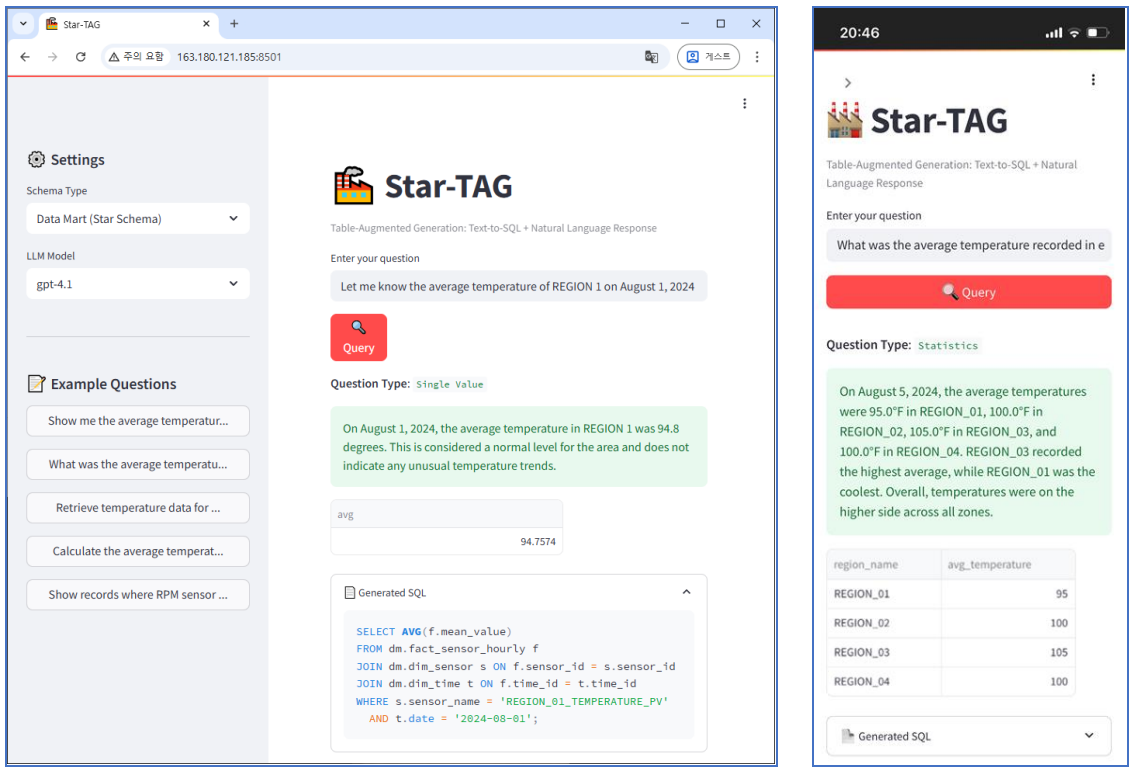
[3)] *Bold indicates the best performance in each row.*

A particularly notable result is that Qwen3, despite having the smallest parameter size (4B), achieved the highest Execution Accuracy (EX) of 75.5% under Star-TAG—comparable to, and marginally exceeding, GPT-4.1 (74.5%). This finding suggests that schema-driven optimization can effectively compensate for model-scale limitations, enabling smaller models to achieve competitive Text-to-SQL performance. In terms of token efficiency, Star-TAG consistently reduced token usage across all models, with reduction rates ranging from 45.7% for Qwen3 to 54.4% for GPT-4.1. This reduction can be attributed to the compact star schema representation, which encodes equivalent semantic information in fewer tokens than the wide table's extensive column listings.

4.3 System Interfaces

To demonstrate the practical deployability of the proposed framework, a web-based user interface was implemented. This interface enables shop-floor operators without SQL expertise to query industrial IoT data using natural language and receive contextualized analytical responses in real time.

Figure 3 presents the system interface on (a) PC web and (b) mobile web platforms. The PC web interface consists of four main components, as shown in Figure 3(a). First, the Settings panel allows users to select the schema type (wide table or data mart) and the LLM model (GPT-4.1, Gemma3, or Qwen3), enabling controlled comparison of schema design and model behavior within a unified interface. Second, the Example Questions panel provides predefined shop-floor–oriented queries. Third, upon user input, the system automatically classifies the question type (e.g., single value, statistics, or time series) and returns a natural language response with contextual interpretation, together with the corresponding query result table. Fourth, the generated SQL query is explicitly displayed, ensuring transparency and traceability of the data retrieval process for system auditing. The mobile web interface was designed to support on-site decision making, reflecting the mobility requirements of shop-floor personnel, as shown in Figure 3(b). This design enables workers to access analytical insights directly at the point of operation without reliance on fixed terminals.



(a) PC web interface with single-value query (b) Mobile interface with statistical query

Figure 3. Web-based user interfaces of the Star-TAG system.

The system supports multiple types of data queries commonly required in shop-floor operations. For single-value queries, the system returns aggregated sensor measurements at specified time points. For example, in response to the question "Let me know the average temperature of REGION 1 on August 1, 2024," the system provides both the numerical value (94.8°C) and an interpretation indicating whether it falls within the normal operating range (see Figure 3(a)). For statistical queries, the system performs comparative analysis across multiple regions or equipment. In response to the question "What was the average temperature recorded in each zone on August 5, 2024?", the system presents average temperatures for each region in a table and provides comparative analysis such as identifying the highest and lowest values across zones (see Figure 3(b)). For time-series queries, the system returns sequential measurements over specified periods to support trend analysis and anomaly detection.

5 Discussion

5.1 Text-to-SQL Errors and Self-Correction

This chapter examines how the structural properties of the star schema influence Text-to-SQL reliability through error analysis and assesses the contribution of schema design to error mitigation.

Across the 564 total experiments on Query set B (2 schemas × 94 synthetic queries × 3 models), the wide-table configuration resulted in 24 execution failures (8.5%), whereas the Star-TAG configuration produced only 4 failures (1.4%). Figure 4 summarizes the distribution of error types across models and schema configurations.

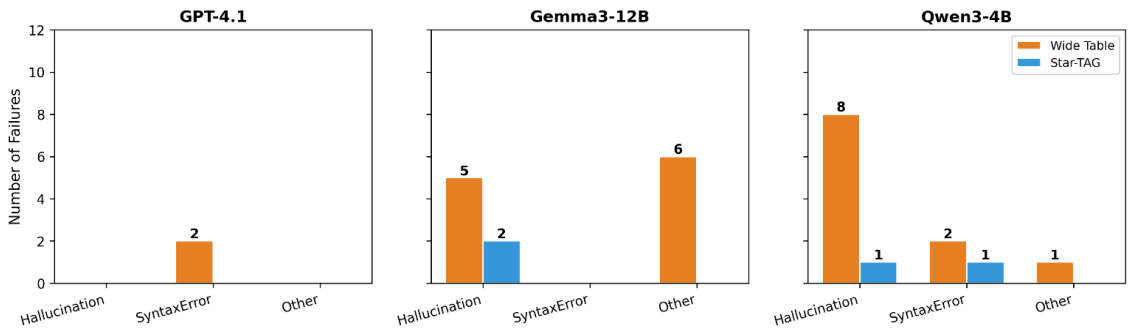
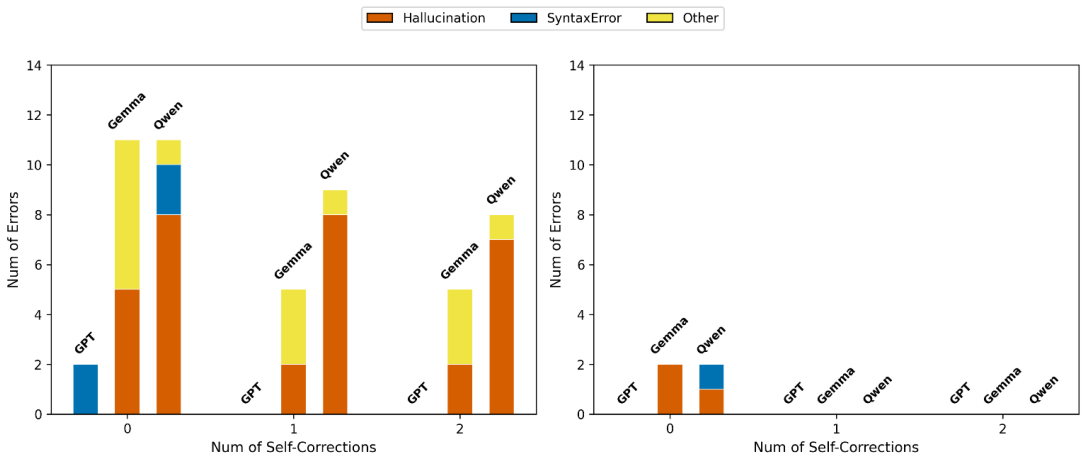


Figure 4. Distribution of Text-to-SQL error types across models and schema configurations.

Hallucination—where the LLM generates references to non-existent columns—was the dominant error type in both schemas; however, it occurred frequently in the wide-table configuration (8 cases for Qwen3-4B and 5 cases for Gemma3-12B), while Star-TAG exhibited substantially fewer occurrences (2 cases for Gemma3-12B and 1 case for Qwen3-4B). Notably, GPT-4.1 exhibited no hallucination errors under either schema configuration, suggesting greater robustness to schema complexity in larger models. These results indicate that the explicit fact–dimension structure of the star schema effectively reduces column selection errors by constraining the LLM's search space, with particularly pronounced benefits for resource-constrained open-source models.

Figure 5 illustrates the effectiveness of self-correction, in which PostgreSQL error messages generated from failed SQL executions were fed back to the LLM to regenerate queries.

Under the Star-TAG configuration, all four initial execution failures were resolved with a single retry, achieving a 100% execution success rate across all models. In contrast, under the wide-table schema, Gemma3-12B and Qwen3-4B retained 5 and 8 execution failures, respectively, even after two self-correction attempts. Only GPT-4.1 achieved complete error resolution with a single retry under the wide-table schema, indicating that higher-capacity models can partially mitigate the impact of schema complexity.



(a) Wide-table configuration (b) Star-TAG configuration

Figure 5. Error counts across self-correction iterations.

This disparity in recovery performance can be attributed to differences in schema complexity. The Star-TAG schema contains only 12 columns in total; therefore, even when hallucination occurs, the correct column name can often be inferred from the schema context provided alongside the database error message (e.g., "column X does not exist"). In contrast, the wide-table schema comprises 69 columns with numerous similarly named fields, making it difficult for the LLM to identify the intended column based on the error message alone. Consequently, Star-TAG not only reduces the frequency of execution errors but also facilitates effective error recovery through self-correction, resulting in higher reliability for practical deployment in manufacturing environments.

5.2 Why Star-TAG Improves Text-to-SQL Performance

This chapter analyzes the structural factors through which the star schema design underlying Star-TAG improves Text-to-SQL performance. Three key mechanisms are identified: temporal expression clarity, output format consistency, and schema complexity reduction. First, the star schema enhances the clarity of temporal expressions. In the wide-table configuration, temporal conditions such as "from 10 AM to 12 PM" can be translated into multiple semantically equivalent but syntactically different SQL formulations, including timestamp range predicates (e.g., "CreDate" >= '2024-08-01 10:00:00' AND "CreDate" < '2024-08-01 12:00:00'), EXTRACT-based conditions (e.g., EXTRACT(HOUR FROM "CreDate") BETWEEN 10 AND 12), or date_trunc-based expressions. This syntactic variability increases the likelihood of mismatch between LLM-generated queries and ground-truth SQL—even when both are semantically correct. In contrast, the star schema provides an explicit and discrete time dimension (e.g., dim_time.hour BETWEEN 10 AND 12), thereby reducing representational ambiguity and guiding the LLM toward a more consistent formulation.

Second, the star schema promotes consistency in output structure. When aggregating values across multiple regions, the wide-table schema permits both horizontal output formats (e.g., SELECT MAX("REGION_01_TEMP"), MAX("REGION_02_TEMP"), ...) and vertical formats constructed via UNION ALL. Such variability can lead to inconsistent result structures. By contrast, the star schema naturally yields normalized, vertical outputs through GROUP BY region_name, which closely aligns with standard analytical query patterns commonly observed during LLM pre-training. This structural alignment reduces uncertainty in query formulation and improves result consistency.

Third, the star schema reduces effective schema complexity. While the wide table requires the LLM to reason over 69 heterogeneous columns simultaneously, the star schema distributes this information across four semantically distinct tables connected by explicit foreign key relationships. This organization allows LLMs to leverage learned fact–dimension join patterns instead of relying on brittle inference over complex column naming conventions. As a result, the schema interpretation burden is reduced, leading to more reliable SQL generation.

6 Conclusions and Future Work

This study proposed a data-centric approach to improve Text-to-SQL performance in manufacturing IoT environments. The core contributions are twofold: (1) a star schema–based data mart design that restructures wide-table IoT data into an LLM-friendly format, and (2) a TAG pipeline optimized for this structure, enabling natural language–based data access for shop-floor operators without SQL expertise.

Experimental results demonstrated that the proposed Star-TAG framework outperformed conventional wide-table–based approaches in terms of accuracy, efficiency, and robustness. Notably, schema optimization enabled resource-constrained open-source models to achieve performance comparable to larger commercial models. These findings suggest that schema design optimization represents a more effective and sustainable performance improvement strategy than prompt engineering or model scaling, offering practical possibilities for deploying on-premise AI systems in manufacturing environments subject to security and computational constraints.

Despite these contributions, this study has several limitations. First, validation was conducted within a single industrial domain (MES) and on a single database platform (PostgreSQL); therefore, generalizability to diverse manufacturing processes and database environments requires further investigation. Second, although qualitative analysis of Interpretive Answer Generation was performed, systematic user studies involving shop-floor operators would strengthen the evaluation of response quality, usability, and decision-support effectiveness.

Future research directions are as follows. First, query clarification mechanisms are required to resolve ambiguity in natural language queries by generating follow-up questions and refining user intent through dialogue. Second, to further enhance accessibility for shop-floor operators, the system can be extended with speech-based interfaces leveraging STT and TTS technologies, supporting hands-free interaction in operational environments. Third, beyond reactive question answering, the framework can be evolved into an autonomous monitoring agent capable of proactively detecting and reporting anomalous patterns in industrial data streams.

References

Biswal A, Patel L, Jha S, Kamsetty A, Liu S, Gonzalez JE, et al. Text2SQL is Not Enough: Unifying AI and Databases with TAG. In: *Conference on Innovative Data Systems Research (CIDR)*; 2025 Jan 19-22; Amsterdam, Netherlands. https://www.vldb.org/cidrdb/papers/2025/p11-biswal.pdf.

Blondheim DJ. The 6C Framework to Build a Connected Factory. *Int J Met* 2025. https://doi.org/10.1007/s40962-025-01674-9.

Brown T, Mann B, Ryder N, Subbiah M, Kaplan JD, Dhariwal P, et al. Language Models are Few-Shot Learners. *Adv Neural Inf Process Syst* 2020;33:1877-901. https://proceedings.neurips.cc/paper/2020/hash/1457c0d6bfcb4967418bfb8ac142f64a-Abstract.html.

Cao Z, Zheng Y, Fan Z, Zhang X, Chen W, Bai X. RSL-SQL: Robust Schema Linking in Text-to-SQL Generation. *arXiv preprint* 2024. https://doi.org/10.48550/arXiv.2411.00073.

Colabianchi S, Costantino F, Sabetta N. Assessment of a large language model based digital intelligent assistant in assembly manufacturing. *Comput Ind* 2024;162:104129. https://doi.org/10.1016/j.compind.2024.104129.

Garcia CI, DiBattista MA, Letelier TA, Halloran HD, Camelio JA. Framework for LLM applications in manufacturing. *Manuf Lett* 2024;41:253-63. https://doi.org/10.1016/j.mfglet.2024.09.030.

Hong Z, Yuan Z, Zhang Q, Chen H, Dong J, Huang F, et al. Next-Generation Database Interfaces: A Survey of LLM-Based Text-to-SQL. *IEEE Trans Knowl Data Eng* 2025;37:7328-45. https://doi.org/10.1109/TKDE.2025.3609486.

Kernan Freire S, Wang C, Foosherian M, Wellsandt S, Ruiz-Arenas S, Niforatos E. Knowledge sharing in manufacturing using LLM-powered tools: user study and model benchmarking. *Front Artif Intell* 2024;7:1293084. https://doi.org/10.3389/frai.2024.1293084.

Kimball R, Ross M. *The Data Warehouse Toolkit: The Definitive Guide to Dimensional Modelin*g. 3rd ed. Hoboken: John Wiley & Sons; 2013.

Kumar N, Lee SC. Human-machine interface in smart factory: A systematic literature review. *Technol Forecast Soc Change* 2022;174:121284. https://doi.org/10.1016/j.techfore.2021.121284.

Lei F, Chen J, Ye Y, Cao R, Shin D, Su H, et al. Spider 2.0: Evaluating Language Models on Real-World Enterprise Text-to-SQL Workflows. In: *International Conference on Learning Representations*; 2025; Singapore.

Lewis P, Perez E, Piktus A, Petroni F, Karpukhin V, Goyal N, et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Adv Neural Inf Process Syst* 2020;33:9459-74. https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html.

Li H, Wu S, Zhang X, Huang X, Zhang J, Jiang F, et al. OmniSQL: Synthesizing High-quality Text-to-SQL Data at Scale. *Proc VLDB Endow* 2025;18:4695-709.

Li J, Hui B, Qu G, Yang J, Li B, Li B, et al. Can LLM Already Serve as A Database Interface? A Big Bench for Large-Scale Database Grounded Text-to-SQLs. *Adv Neural Inf Process Syst* 2023;36. https://proceedings.neurips.cc/paper_files/paper/2023/hash/83fc8fab1710363050bbd1d4b8cc0021-Abstract-Datasets_and_Benchmarks.html.

Ma K, Yang B. Multiple Wide Tables with Vertical Scalability in Multitenant Sensor Cloud Systems. *Int J Distrib Sens Netw* 2014;10:583686. https://doi.org/10.1155/2014/583686.

Maedche A, Legner C, Benlian A, Berger B, Gimpel H, Hess T, et al. AI-Based Digital Assistants. *Bus Inf Syst Eng* 2019;61:535-44. https://doi.org/10.1007/s12599-019-00600-8.

McTear M. *Conversational AI: Dialogue Systems, Conversational Agents, and Chatbots*. Cham: Springer Nature; 2022. https://doi.org/10.1007/978-3-031-02176-3.

Mourtzis D, Angelopoulos J, Panopoulos N. Operator 5.0: A Survey on Enabling Technologies and a Framework for Digital Manufacturing Based on Extended Reality. *J Mach Eng* 2022;22:43-69. https://doi.org/10.36897/jme/147160.

Qu G, Li J, Li B, Qin B, Huo N, Ma C, et al. Before Generation, Align it! A Novel and Effective Strategy for Mitigating Hallucinations in Text-to-SQL Generation. In: *Findings of the Association for Computational Linguistics*; 2024; Bangkok, Thailand. p. 5456-71. https://doi.org/10.18653/v1/2024.findings-acl.324.

Tao F, Qi Q, Wang L, Nee AYC. Digital Twins and Cyber-Physical Systems toward Smart Manufacturing and Industry 4.0: Correlation and Comparison. *Engineering* 2019;5:653-61.

https://doi.org/10.1016/j.eng.2019.01.014.

Wang C, Qiao J, Huang X, Song S, Hou H, Jiang T, et al. Apache IoTDB: A Time Series Database for IoT Applications. *Proc ACM Manag Data* 2023;1:1-27. https://doi.org/10.1145/3589775.

Wei J, Wang X, Schuurmans D, Bosma M, Ichter B, Xia F, et al. Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. *Adv Neural Inf Process Syst* 2022;35:24824-37. https://proceedings.neurips.cc/paper_files/paper/2022/hash/9d5609613524ecf4f15af0f7b31abca4-Abstract-Conference.html.

Wellsandt S, Klein K, Hribernik K, Lewandowski M, Bousdekis A, Mentzas G, et al. Hybrid-augmented intelligence in predictive maintenance with digital intelligent assistants. *Annu Rev Control* 2022;53:382-90. https://doi.org/10.1016/j.arcontrol.2022.04.001.

Yu T, Zhang R, Yang K, Yasunaga M, Wang D, Li Z, et al. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In: *Conference on Empirical Methods in Natural Language Processing*; 2018; Brussels, Belgium. p. 3911-21. https://doi.org/10.18653/v1/D18-1425.

Yuan Z, Li M, Liu C, Han F, Huang H, Dai HN. Chat with MES: LLM-driven user interface for manipulating garment manufacturing system through natural language. *J Manuf Syst* 2025;80:1093-107. https://doi.org/10.1016/j.jmsy.2025.02.008.

Zha D, Bhat ZP, Lai KH, Yang F, Jiang Z, Zhong S, et al. Data-centric Artificial Intelligence: A Survey. *ACM Comput Surv* 2025;57:129:1-129:42. https://doi.org/10.1145/3711118.

Zhang C, Xu Q, Yu Y, Zhou G, Zeng K, Chang F, et al. A survey on potentials, pathways and challenges of large language models in new-generation intelligent manufacturing. *Robot Comput Integr Manuf* 2025;92:102883. https://doi.org/10.1016/j.rcim.2024.102883.

Zhong RY, Xu X, Klotz E, Newman ST. Intelligent Manufacturing in the Context of Industry 4.0: A Review. *Engineering* 2017;3:616-30. https://doi.org/10.1016/j.eng.2017.05.015.