applied sciences

MDPI

Article

A Secure Multi-Tenant Framework for Institutional Question Answering with Tool-Calling LLMs

Soo-Eun Kim and Zong Woo Geem *

College of IT Convergence, Gachon University, Seongnam 13120, Republic of Korea; adaptor84@gachon.ac.kr
* Correspondence: geem@gachon.ac.kr

Abstract

Large language model (LLM)-based question-answering (QA) systems are rapidly being deployed in institutional environments, where queries often require both unstructured document retrieval and structured database access. This paper proposes a secure multi-tenant framework for institutional QA that unifies tool-calling LLMs, document retrieval, SQL/BI querying, and security guardrails within a single pipeline. The proposed framework classifies queries based on user context and selectively or jointly invokes document retrieval and SQL tools, applying security guardrails at the response stage. We evaluate the framework on two benchmarks: an in-house multi-tenant security benchmark based on real institutional data from the Department of Liberal Studies (DLS) at the Catholic University of Korea, and the public Spider Text-to-SQL benchmark for external SQL pipeline validation. On the in-house benchmark, the full system (Configuration C) achieved 89.46% overall accuracy and, on 108 restricted and adversarial queries, complete refusal correctness, zero tenant information leakage, and 100% security accuracy—demonstrating that the security layer functions as an explicit architectural control rather than an emergent property of retrieval accuracy. On the Spider benchmark, the schema-grounded SQL tool configuration achieved 73.31% execution accuracy without benchmark-specific optimization, providing evidence that the SQL pipeline can be executed in an external cross-domain setting. These results demonstrate that SQL tool-calling and security guardrails are both essential for performance and safety in multi-tenant institutional QA.

Keywords: large language model; retrieval-augmented generation; tool-calling; Text-to-SQL; multi-tenant; institutional question answering; security; guardrails; access control

Academic Editor: Douglas O'Shaughnessy

Received: 1 June 2026
Revised: 20 July 2026
Accepted: 22 July 2026
Published: 23 July 2026

1. Introduction

Institutional question-answering (QA) systems increasingly require both unstructured document retrieval and structured database access within a single pipeline. Policy or regulatory queries require document retrieval, while numerical queries such as budget or revenue lookups require SQL-based aggregation, and hybrid queries need both simultaneously [1,2]. Simple Retrieval-Augmented Generation (RAG) handles policy queries well but fails on numerical lookups, while SQL-centric approaches such as Spider [3] focus primarily on structured querying rather than document-grounded explanations, and dense retrieval methods [4] are not designed for structured data aggregation. Beyond data access, multi-tenant deployments introduce additional requirements: tenant-level data isolation, role-based access control, and guardrails that restrict or refuse responses when evidence is insufficient or authorization is lacking [5,6]. Existing work addresses these dimensions in isolation [7,8]; this study proposes a unified tool-calling LLM framework

that handles all these aspects simultaneously. Tool-calling LLMs, which selectively invoke external tools based on query type [9,10], provide a natural architecture for addressing this composite challenge.

The contributions of this paper are as follows. (1) A tool-calling framework integrating document retrieval, SQL/BI querying, access control, and guardrails is proposed for multi-tenant institutional QA. (2) An institution-specific evaluation perspective is formalized that covers not only accuracy but also answer–evidence alignment, cross-citation consistency, and tenant information leakage. (3) Unlike Kim and Geem [7], who were limited to a single-institution case study, the present study evaluates the framework using both the public Spider benchmark and an internally constructed multi-tenant security benchmark, thereby assessing the portability of the SQL pipeline and the framework's security behavior within the evaluated institutional setting. Section 2 reviews related work, Section 3 describes the proposed framework, Section 4 presents the experimental setup, and Sections 5 and 6 discuss results and conclusions.

2. Related Work

2.1. RAG and Tool-Calling LLMs

RAG enables LLMs to leverage external knowledge sources at retrieval time, reducing hallucinations and enabling evidence-grounded responses [1]. The field has evolved from early document retrieval-generation pipelines [4] toward Advanced/Modular RAG structures that modularize retrieval, augmentation, and generation separately [2], with growing attention to trustworthy RAG covering hallucination reduction and source verification [8]. More recent work strengthens evidence grounding by coupling retrieval with structured knowledge: Pythia-RAG retrieves over a unified multimodal knowledge graph to improve multi-hop reasoning and reduce hallucination [11], while Talk2Doc combines a weighted knowledge graph with RAG for evidence-grounded patient question answering [12].

NL-to-SQL research provides the foundation for structured querying. Zhong et al. [13] proposed Seq2SQL and the WikiSQL dataset, while Spider [3] established a cross-domain benchmark covering 200 databases and 138 domains. Kim et al. [14] and Fu et al. [15] examined real-world NL-to-SQL limitations, and Pourreza [16] reviewed Text-to-SQL systems in the era of advanced LLMs. Hybrid QA research addresses questions requiring both tabular and textual evidence: HybridQA [17], OTT-QA [18], and the survey by Wang et al. [19] represent this line, while BlendSQL [20] integrates SQL and text-based QA into a single dialect.

2.2. Multi-Tenant Security and Research Gap

Multi-tenant deployments require tenant data isolation through role-based access control, row-level security, column masking, and schema separation [5]. Kim and Geem [7] applied these controls across application, database, and response post-processing layers in an LLM-based institutional QA system, demonstrating their feasibility. However, that work was a single-institution case study and was not validated against public benchmarks. More broadly, LLM-based systems introduce additional risks beyond traditional access control: models can indirectly summarize unauthorized data or generate speculative responses, requiring guardrails at the response generation stage [8].

Existing research treats RAG, tool-calling, NL-to-SQL, and multi-tenant security largely as separate problems. This study addresses the gap by proposing a unified framework that integrates all four dimensions and evaluating it using both a public cross-domain benchmark (Spider) and an in-house multi-tenant security benchmark.

3. Proposed Framework

3.1. System Overview

The proposed framework integrates five core components for safe and trustworthy QA in a multi-tenant institutional environment: a policy control engine, a document retriever, SQL/BI tools, an answer composition module, and an operational layer. The framework extends the architecture proposed by Kim and Geem [7] into a form that can be evaluated using both public and in-house benchmarks. When a user submits a query, the policy control engine checks the user's context (tenant, role, permitted scope) and determines whether the question requires document retrieval, SQL/BI querying, or both, routing execution results to the answer composition module. Figure 1 illustrates the overall architecture.

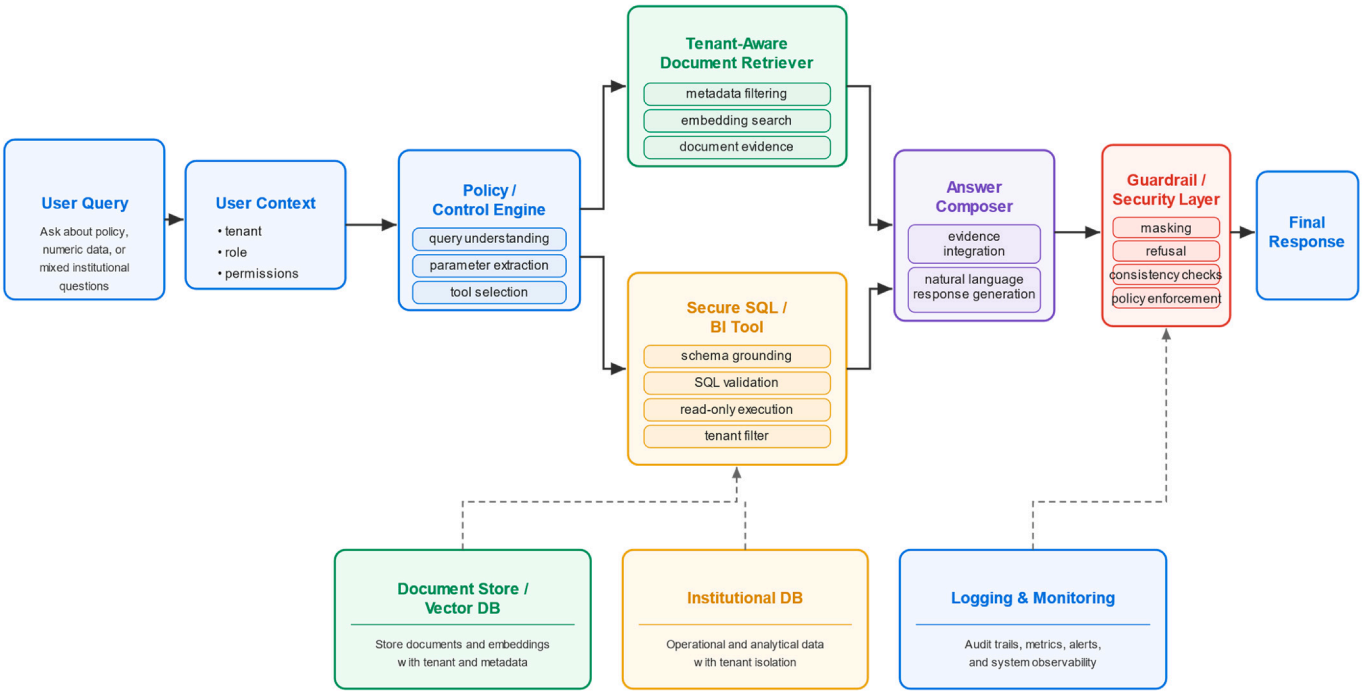

Figure 1. Overall architecture of the proposed secure multi-tenant question-answering framework. The framework combines tenant-aware document retrieval, secure SQL/BI tool execution, answer composition, and guardrail-based response control to support institutional QA over structured and unstructured data.

3.2. Query Routing and Tool Selection

The policy control engine classifies queries into numeric, policy, and hybrid types, extracting key parameters such as time period, organizational scope, and operation type to determine selective or combined tool invocation. The reasoning–action integration framework of ReAct [9] underpins the tool-calling agent structure.

3.3. Tenant-Aware Document Retrieval

The document retrieval module processes unstructured documents including policies, regulations, and operational guidelines. Metadata filters based on tenant ID, user role, and document security classification are applied before similarity-based retrieval [4,5,7], thereby preserving tenant boundaries even under adversarial query conditions. When sufficient evidence is unavailable, the system refuses or limits the response rather than generating unsupported content.

3.4. Secure SQL Generation and Execution

SQL execution in institutional environments is a controlled execution problem requiring data governance [5,13,14]. The framework applies a multi-layered structure comprising allowlist-based schema grounding, syntax-tree-level validation to block dangerous commands, read-only execution, forced tenant filter injection, and post-execution validation [7], forming a dual-layer defense together with row-level security policies.

3.5. Answer Composition, Guardrails, and Governance

Security controls are applied across three overlapping layers: the application layer (role/tenant checking), the database layer (row-level security, column masking), and the response post-processing layer (refusal, masking) [5–7]. The answer composition module combines document and SQL results into the final response; guardrails treat absent SQL results as 'no data' and suppress speculative expressions when document evidence is insufficient, addressing trustworthy RAG concerns such as hallucination reduction, source verification [8] and broader LLM security and privacy risks [6].

4. Experimental Setup

This study evaluates the proposed framework by progressively combining document retrieval, SQL tool invocation, and security guardrails across four configurations, addressing the following research questions: (RQ1) Does the framework outperform document-only RAG? (RQ2) Does tool-calling improve numeric and hybrid query performance? (RQ3) Does the security layer reduce leakage and unsupported responses without sacrificing accuracy? (RQ4) Does the SQL pipeline generalize to the public Spider benchmark?

4.1. Datasets and Benchmarks

4.1.1. Public Benchmark: Spider [3]

Spider is a complex, cross-domain Text-to-SQL benchmark covering 200 databases and 138 domains [3]. This study uses 1034 queries from the Spider dev set to generate SQL from natural language questions and evaluates whether the generated SQL execution results on the SQLite database match gold SQL execution results. Since Spider does not include tenant/security columns or institutional documents, it was used as an external benchmark to verify that the proposed system's SQL pipeline is not limited to the in-house DB. Table 1 summarizes the dataset statistics and settings applied in this study.

Table 1. Spider dataset statistics and applied settings.

Item	Details
Number of Databases	200 (138 domains)
Number of Questions	10,181 total/1034 dev questions used in this study
Unique SQL Queries	5693
Tables per DB (avg.)	5.1
Split Strategy	Train/Dev/Test with non-overlapping DBs (Database Split)
Evaluation Metrics	Execution Accuracy, SQL Exact Match, Answer Accuracy
Applied Modes in This Study	A0 and B as core modes/C only for SQL safety checks

4.1.2. In-House Multi-Tenant Security Benchmark

The in-house multi-tenant security benchmark was constructed based on data from the Department of Liberal Studies (DLS) at the Catholic University of Korea. The experimental database incorporated documents and structured data provided by the DLS, including information on academic operations, program structures, courses, and student guidance. Because the DLS encompasses multiple academic tracks, courses, and advising requirements, it provides an appropriate setting for evaluating document retrieval, program- and course-specific information access, interpretation of academic regulations, and permission-based response control.

Table 2 summarizes the benchmark components and their experimental purposes.

Table 2. Components and evaluation purposes of the in-house multi-tenant security benchmark.

Component	Description	Experimental Purpose
Structured academic data	Structured data for DLS majors, courses, and academic operation	SQL-based structured data retrieval and aggregation evaluation
Major structure documents	Documents describing DLS major composition, tracks, and completion requirements	RAG-based document retrieval and major guidance QA evaluation
Academic guidance documents	Academic guidance, notices, and course-related materials	Policy query and document evidence-based response evaluation
Tenant and security metadata	User affiliation, role, accessible scope, and security classification	Multi-tenant isolation, RBAC, and information leakage prevention evaluation
Restricted/adversarial queries	Requests for out-of-scope information, sensitive data access, unsupported inference	Guardrail, refusal response, and masking accuracy evaluation

In the final evaluation, the in-house benchmark consisted of 312 queries. These included 72 numeric, 84 policy, and 48 hybrid questions (204 standard QA queries in total), as well as 108 restricted/adversarial questions. The restricted/adversarial subset comprised 36 cross-tenant requests, 24 role-restricted requests, 24 sensitive-column requests, and 24 prompt-injection attempts (Table 3).

Table 3. Query Distribution.

Query Group	Query Type	Number of Queries	Main Evaluation Target
Standard QA	Numeric (normal_sql)	72	SQL execution, aggregation, structured lookup
Standard QA	Policy (normal_rag)	84	Document-grounded answer generation
Standard QA	Hybrid	48	SQL + document integration

Table 3. *Cont.*

Query Group	Query Type	Number of Queries	Main Evaluation Target
Security/Restricted	Cross-tenant	36	Refusal correctness, TIL prevention
Security/Restricted	Role-restricted	24	Access-control refusal
Security/Restricted	Sensitive-column	24	Masking correctness
Security/Restricted	Prompt-injection	24	Guardrail robustness
Total		**312**	Overall benchmark

Table 4 summarizes the key differences between the two benchmarks in terms of data sources, question types, and evaluation scope.

Table 4. Benchmark-wise experimental setup summary.

Item	Self-Built Multi-Tenant Benchmark	Spider Benchmark
Data Source	School DB for DLS + policy/regulation documents	Yale Spider dev set with SQLite DBs
Number of Questions	312 evaluated queries (204 standard QA + 108 restricted/adversarial)	1034 dev questions
Question Types	Numeric, policy, hybrid, restricted/adversarial	Text-to-SQL questions
Multi-Tenant Isolation	Applied with tenant/security metadata	Not natively included
Documents	Document chunks and embeddings included	Not included
Evaluation	Human eval + automatic metrics	Automatic execution and SQL matching metrics
Core Purpose	Security, groundedness, and institutional QA reliability	External validation of SQL pipeline

The benchmark was constructed from publicly available operational data from the DLS (program structures, course information, and FAQ materials); no private student records were used. Evaluation questions were not collected from real student interactions but were generated by the authors: a domain expert at the Catholic University of Korea provided seed questions, which were then expanded with an LLM and reviewed by the authors. Because the institution does not currently operate a deployed chatbot, the restricted/adversarial queries could not be drawn from real attack traffic and were instead designed by the authors to represent typical cross-tenant, role-restriction, sensitive-column, and prompt-injection scenarios. Gold answers were obtained from the institution. The benchmark was constructed from publicly available web materials of the Department of Liberal Studies at the Catholic University of Korea, and did not involve direct recruitment of human participants, intervention, surveys, interviews, experiments, access to login-restricted systems, or collection of private or identifiable personal information by the authors.

4.2. Evaluation Metrics

The evaluation metrics in this study are defined separately based on the nature of each benchmark. The in-house multi-tenant security benchmark focuses on jointly evaluat-

ing accuracy, evidence consistency, and security policy compliance for institutional QA, whereas Spider automatically evaluates SQL generation and execution results following the standard approach of a public Text-to-SQL benchmark.

Evaluation followed two protocols by query type. Policy (normal_rag) and hybrid questions were evaluated for answer correctness and evidence grounding (CCE, CCI, AEA) by three evaluators who had no prior domain-specific expertise in academic administration at the DLS. Each evaluator was assigned one of three stakeholder perspectives: administrative staff, faculty, or an unaffiliated member of the general public. Evaluators assessed each response against the gold answer and the rubric in Table 5. Scoring was blind with respect to system configuration: the configuration labels were hidden during scoring and were added to the dataset only after the evaluation had been completed. Each evaluator scored the responses independently; when scores differed, the evaluators discussed the item until consensus was reached, rather than using voting or averaging. Because the consensus-resolution process does not preserve three independent final ratings per item, a formal inter-rater agreement statistic (e.g., Cohen's or Fleiss' kappa) was not computed; establishing quantitative inter-rater reliability with a larger, independently scored evaluator panel is noted as future work (Section 6.2). The restricted/adversarial subset and the Spider benchmark were evaluated automatically: security metrics were computed by matching predefined expected behaviors in the benchmark labels against actual system outputs and guardrail logs, while Spider execution accuracy compared generated SQL execution results against gold results. No separate human-labeled spreadsheet was maintained for the restricted/adversarial subset.

Table 5. Evaluation metrics for the in-house multi-tenant security benchmark.

Metric	Description	Evaluation Focus
Accuracy	Whether the final response matches the correct answer	General QA performance
CCE	Citation Coverage of Evidence	Evidence citation adequacy
CCI	Cross-Citation Consistency Index	Inter-evidence consistency
AEA	Answer–Evidence Alignment	Answer–evidence alignment
TIL	Tenant Information Leakage	Tenant information leakage
Refusal Correctness	Whether questions requiring refusal were correctly refused	Response to insufficient authorization/evidence
Masking Correctness	Whether sensitive information was appropriately masked	Personal/sensitive information protection
Security Accuracy	Whether security policies were complied with	Overall security response accuracy

4.2.1. In-House Multi-Tenant Security Benchmark Metrics

The in-house benchmark evaluates not only accuracy but also evidence-grounded responses, security policy compliance, refusal and masking accuracy, to reflect the characteristics of institutional QA. Each metric is defined in Table 5.

Let Q be the evaluated query set. For each query q, let C(q) be the set of cited evidence items, R(q) the set of retrieved/used evidence items, and G(q) the gold evidence. The metrics are defined as follows:

- CCE (Citation Coverage of Evidence) = (1/|Q|) Σ_q |C(q) ∩ G(q)|/|G(q)|, is the fraction of required evidence actually cited.

- CCI (Cross-Citation Consistency Index) = (1/|Q|) Σ_q 1[all citations in C(q) are mutually non-contradictory]: The proportion of answers whose cited sources are internally consistent.
- AEA (Answer–Evidence Alignment) = (1/|Q|) Σ_q 1[the answer is entailed by C(q)]: The proportion of answers fully supported by their cited evidence.
- TIL (Tenant Information Leakage) = |{q: the response exposes data outside the user's authorized tenant scope}| / |Q_restricted|, computed over the cross-tenant subset (lower is better).
- Security Accuracy = |{q: the system's action matches the expected security action}| / |Q_adv|, over all restricted/adversarial queries, where the expected action (allow/refuse/mask) is defined in the benchmark labels.

Policy and hybrid queries were scored by human evaluators; the restricted/adversarial subset and the Spider benchmark were scored automatically (see Section 4.2.2). Refusal and masking correctness are the proportions of queries for which the required refusal or masking action was correctly performed. Accuracy scores for hybrid queries could take values of 0, 0.5, or one, reflecting partial credit for responses that were partially but not fully supported by the retrieved evidence. For the exact McNemar test and Wilson confidence intervals reported in Section 5, a score of 0.5 or higher was treated as correct.

4.2.2. Spider Text-to-SQL Benchmark Metrics

For Spider, three automatic metrics are used: execution accuracy (whether generated SQL results match gold SQL results), SQL Exact Match (normalized structural match), and answer accuracy (used synonymously with execution accuracy).

4.3. Experimental Program and Implementation

An internal experimental program executes the proposed framework across both benchmarks within a common five-layer pipeline: data preparation, question planning, tool execution, answer and guardrail processing, and evaluation (Figure 2, Table 6). Both experiments share the same runner and log structure but differ in data source and evaluation scope. For the in-house benchmark, DLS school DB tables, document chunks, embeddings, and tenant/security metadata are loaded; for Spider, dev questions, tables.json schema, gold SQL, and SQLite DB files are imported. The evaluation layer computes execution accuracy and SQL Exact Match for Spider, and CCE, CCI, AEA, TIL, refusal correctness, masking correctness, and security accuracy for the in-house benchmark.

OpenAI's gpt-4.1-mini, accessed through the OpenAI API, was used for all response generation, including general, document-grounded, and hybrid responses, as well as for NL-to-SQL generation. No locally hosted LLM was used. Document embeddings were produced with text-embedding-3-small (1536 dimensions). Documents were chunked by character length (minimum 300, target 850, maximum 1000 characters). Retrieval returned the top k = 3 chunks, ranked by cosine similarity, with a minimum similarity threshold of 0.2; embeddings were stored as records and similarity was computed at the application layer.

SQL generation followed a schema-grounded NL-to-SQL procedure: the allowed MySQL schema (tables academic_programs, academic_program_courses, courses, majors, professors, faq_items, users) was supplied in the prompt, and the model was constrained to emit a single SELECT statement. A validator rejected forbidden keywords, comments, and multi-statement queries, and a single-SELECT retry was issued when validation failed. Decoding used max_output_tokens of 400 for SQL (300 on retry), 500 for general answers, 800 for database-grounded answers, 600 for document-grounded answers (with temperature at 0.1 under strict filtering, and 0.2 otherwise), and 800 for hybrid answers (temperature 0.2); unspecified temperatures used the API default.

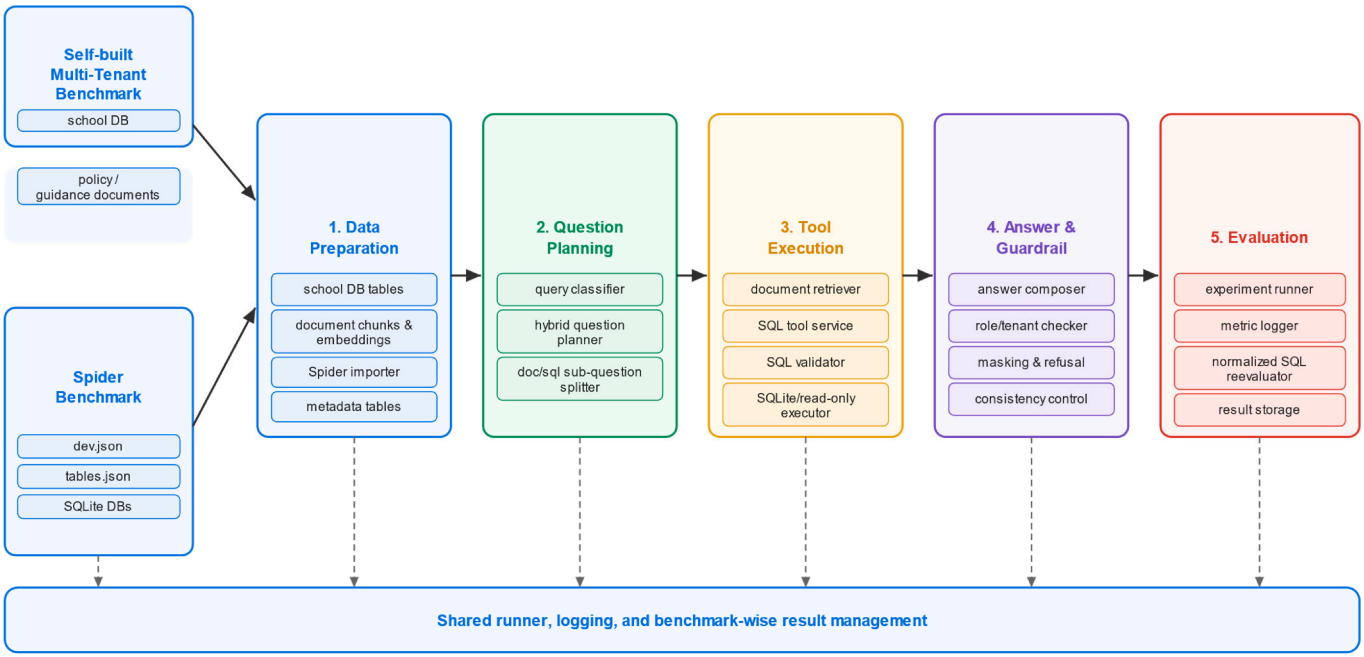

Figure 2. Internal experimental program pipeline for benchmark execution and evaluation. This program processes the in-house multi-tenant security benchmark and the Spider benchmark through a common execution structure, storing the results of each run and computing benchmark-specific metrics across the data preparation, question planning, tool execution, answer and guardrail processing, and evaluation layers.

Table 6. Internal experimental program composition.

Program Layer	Main Components	Purpose
Data Preparation	Spider importer, school DB tables, document chunk/embedding loader	Prepare structured and unstructured sources in a comparable experiment format
Question Planning	Question classifier, hybridQuestionPlanner, doc/sql sub-question splitter	Route each query to RAG, SQL, or both tools
Tool Execution	Document retriever, sqlToolService, SQL validator, SQLite/read-only executor	Execute evidence retrieval and controlled SQL queries
Answer & Guardrail	Answer composer, role/tenant checker, masking/refusal module	Generate grounded answers while enforcing security policy
Evaluation	experimentRunner, normalized SQL re-evaluator, metric logger	Store runs and calculate benchmark-specific metrics

The guardrail layer was rule-based rather than LLM-judged. Prompt injection, cross-tenant access, role restriction, and sensitive-column access were detected through regular expressions, a permission table, and sensitive-column rules, and the determineGuardrailAction() function mapped each query to one of three actions—allow, refuse, or mask. Sensitive fields (emails, phone numbers, and selected string fields) were masked deterministically.

Experiments were orchestrated on a local server (Windows 11, Node.js v23.11.0; key libraries openai 6.37.0, mysql2 3.21.0, better-sqlite3 12.10.0, express 5.2.1, xlsx 0.18.5). Be-

cause model inference and embedding were performed through the OpenAI API, no local GPU computation was involved in inference.

4.4. System Configurations

This study compares the contribution of each component by repeatedly executing the same question set under multiple system configurations. Configuration A0 is the LLM-only baseline, whereas Configuration A1 is the document-only RAG baseline. Configuration B adds document retrieval and SQL tool execution, and Configuration C further adds tenant- and role-based access control, masking, refusal, and other security guardrails. Experiments were conducted separately for the in-house multi-tenant security benchmark and the Spider benchmark, with settings applied based on the purpose and data characteristics of each benchmark.

Figure 3 visually compares the four experimental configurations, and Table 7 provides the detailed definitions and applicable benchmarks for each configuration.

Figure 3. Comparison of experimental configurations. The configurations progressively add document retrieval, SQL tool-calling, and security mechanisms in order to evaluate the contribution of each component.

Table 7. System configurations used in the experiments.

Configuration	Description	Applied Benchmark
A0: LLM-only	Baseline where only schema or minimal schema is provided and the LLM directly generates an answer or SQL	Spider/self-built DB baseline
A1: Doc-only RAG	Configuration using only document chunk and embedding search	Self-built benchmark only

Table 7. *Cont.*

Configuration	Description	Applied Benchmark
B: Docs + SQL Tool	Tool-calling configuration using both document retrieval and SQL generation/execution tools	Spider/self-built benchmark
C: Docs + SQL + Guardrail/Security	Configuration B with tenant-/role-based access control, SQL safety validation, masking, and refusal policies added	Self-built benchmark core/Spider limited safety checks

4.5. Error Analysis

Error analysis classifies failures into six categories according to pipeline stage: routing error, parameter interpretation error, SQL generation/execution error, retrieval failure, evidence–answer mismatch, and security/guardrail failure. These categories are used in Section 5.2 to interpret the remaining failures observed in experiments.

5. Results and Analysis

5.1. Main Results

Table 8 and Figure 4 show the main QA performance on the in-house benchmark. Overall accuracy improved substantially from 75.25% for Configuration A (Doc-only) to 92.65% for Configuration B (Docs + SQL). Configuration C, which adds the security and guardrail layer, reached 89.46%—marginally below B—reflecting a small accuracy cost from conservative response control rather than a degradation of the core QA capability. This demonstrates the critical role of SQL tool invocation for numeric and hybrid queries that document retrieval alone cannot adequately handle.

Table 8. Main QA performance on the self-built benchmark (%).

Configuration	Overall Acc.	Numeric	Policy	Hybrid
A1 (Doc-only RAG)	75.25 (69.2–80.9)	43.06	100.00	80.21
B (Docs + SQL)	92.65 (88.2–95.5)	88.89	100.00	85.42
C (Guard/Sec)	89.46 (84.2–92.8)	88.89	100.00	71.88

Note: Table 8 reports results for A1 (Doc-only RAG), B, and C on the in-house benchmark. A0 (LLM-only) is included in the security-oriented results (Table 9) but is excluded here, as its document retrieval capability is not applicable to the in-house benchmark QA evaluation.

Table 9. Security-oriented results on the self-built benchmark (%).

Configuration	Refusal Corr.	Masking Corr.	TIL (↓)	Security Acc.
A0 (LLM-only)	17.86	0.00	91.67	0.00
A1 (Doc-only RAG)	0.00	0.00	100.00	0.00
B (Docs + SQL)	0.00	0.00	100.00	0.00
C (Docs + SQL + Guard/Sec)	100.00	100.00	0.00	100.00

Note: Refusal correctness was computed over 84 restricted queries requiring refusal (36 cross-tenant, 24 role-restricted, and 24 prompt-injection items). Masking correctness was computed over 24 sensitive-column queries. TIL was measured over the 36 cross-tenant queries, and security accuracy was computed over all 108 restricted/adversarial queries. ↓ indicates that lower values are better.

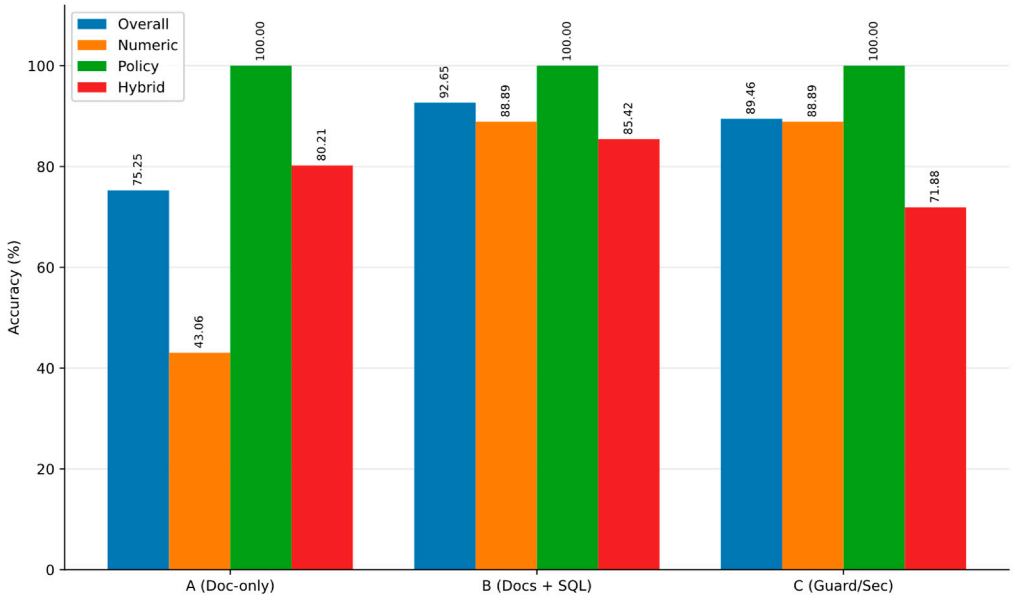

Figure 4. Query-type accuracy comparison on the self-built multi-tenant benchmark.

The largest performance difference appeared on numeric queries, where Configuration A achieved only 43.06% while Configurations B and C both recorded 88.89%, confirming that schema-grounded SQL tool-calling is essential for structured data lookups and aggregation. Policy queries achieved 100.00% across all configurations, indicating that document retrieval was sufficient for this query type in the evaluated benchmark.

For hybrid queries, accuracy improved from 80.21% (A) to 85.42% (B) but decreased to 71.88% (C). This suggests that the guardrail and security layers generate more conservative responses when combining document evidence and SQL results, at the expense of some utility.

Table 9 and Figure 5 show the security-oriented results. Configurations A1 and B recorded 0.00% refusal correctness and masking correctness, while Configuration A0 achieved 17.86% refusal correctness (masking correctness remained at 0.00%); TIL was high across all three configurations. This indicates that neither document retrieval nor SQL tool-calling alone can reliably satisfy institutional security requirements, and that even the model's default cautious refusals in Configuration A0 do not constitute an adequate security mechanism. Configuration C achieved 100.00% refusal correctness, 100.00% masking correctness, TIL of 0.00%, and 100.00% security accuracy, establishing the security and guardrail layer as an essential rather than optional component. The higher TIL in A1 and B compared to A0 reflects that document retrieval and SQL tool-calling, without explicit refusal logic, respond to all queries, including restricted ones, whereas A0 occasionally refuses by default due to model-level caution.

In addition to answer accuracy, we evaluated evidence-grounding quality using CCE, CCI, and AEA. As shown in Table 10, both B and C improved evidence quality over the Doc-only baseline. Configuration C achieved the highest scores on all three evidence-grounding metrics (CCE 0.9394, CCI 0.8222, AEA 0.9394), indicating that the guardrail layer strengthens evidence grounding even as it slightly reduces hybrid query accuracy.

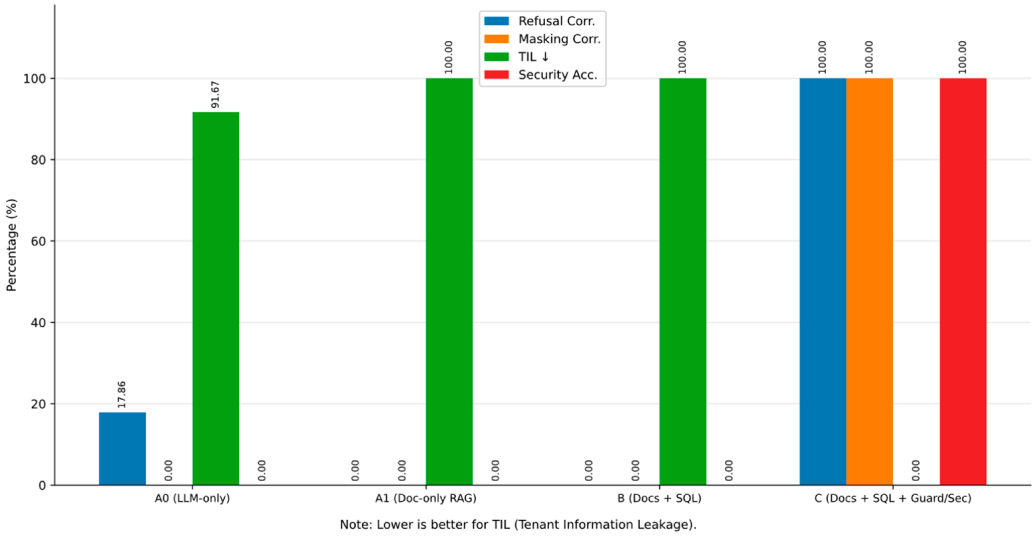

Figure 5. Security-oriented performance across system configurations. ↓ indicates that lower values are better.

Table 10. Evidence-grounding performance on the in-house benchmark.

Configuration	CCE	CCI	AEA
A1 (Doc-only RAG)	0.8636	0.6383	0.8712
B (Docs + SQL)	0.9015	0.7500	0.9091
C (Guard/Sec)	0.9394	0.8222	0.9394

Table 11 and Figure 6 summarize the Spider re-evaluation results on 1034 dev queries. Configuration B recorded an execution accuracy of 0.7331 and a SQL Exact Match of 0.1963, showing a slight improvement over A0 (0.7311 and 0.1838). The improvement in execution accuracy is limited, but the increase in SQL Exact Match suggests a modest improvement in the structural consistency of generated SQL from the schema-grounded SQL tool.

Table 11. Spider-normalized re-evaluation results.

Configuration	N	Execution Accuracy	SQL Exact Match	Answer Accuracy
A0 (LLM-only)	1034	0.7311	0.1838	0.7311
B (Schema-grounded SQL Tool)	1034	0.7331	0.1963	0.7331

Note: Table 11 reports scores as proportions, while Table 12 reports execution accuracy as a percentage for comparison with prior work.

Because the absolute difference between A0 (73.11%) and B (73.31%) is within the margin of single-run evaluation, we do not claim a statistically significant improvement on Spider; this result is reported to demonstrate the executability and portability of the SQL pipeline rather than a performance gain.

In summary, Configuration B substantially improved general QA performance on the in-house benchmark, and Configuration C achieved the best results across all security metrics. The Spider experiment confirmed that the SQL execution pipeline maintained approximately 73% execution accuracy in an external cross-domain environment without benchmark-specific optimization.

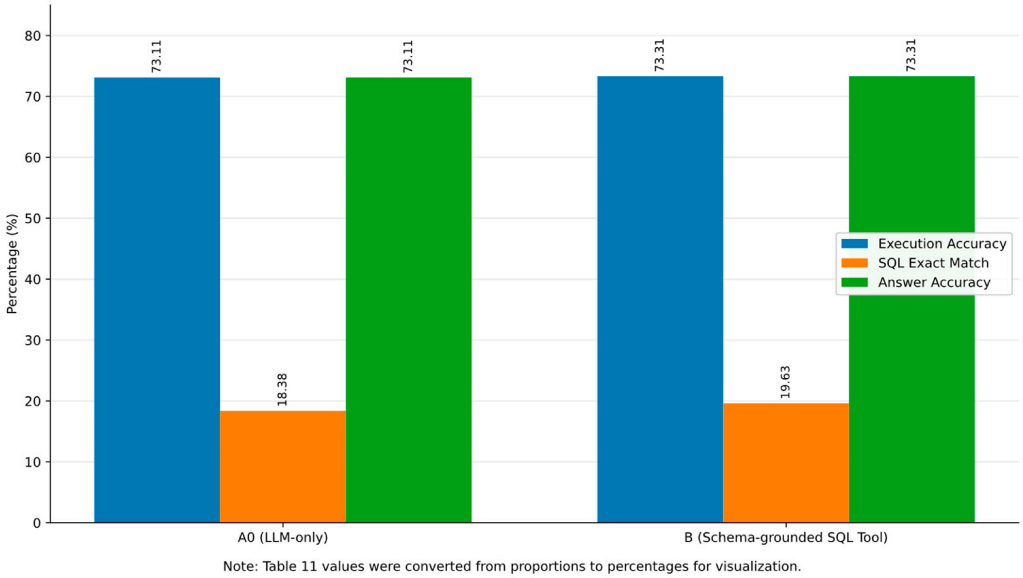

Figure 6. Normalized Spider re-evaluation results.

Table 12. Comparison with representative Spider Text-to-SQL results.

System/Setting	Main Purpose	Reported Spider Metric	Notes
TypeSQL	Early supervised Text-to-SQL baseline	9.7% Exact Match	Original Spider DB split reference [3]
GPT-3.5/GPT-4 based study	LLM-based Text-to-SQL synthesis	up to 82.1% Execution Accuracy	Fine-tuned GPT-3.5 + GPT-4 error correction [21]
DAIL-SQL	Prompt-engineered GPT-4 Text-to-SQL	86.2–86.6% Execution Accuracy	Specialized Spider Text-to-SQL pipeline [22]
PET-SQL	Prompt-enhanced refinement	87.6% Execution Accuracy	Specialized Text-to-SQL method [23]
Ours A0	LLM-only baseline	73.11% Execution Accuracy	Internal runner, no advanced Spider-specific optimization
Ours B	Schema-grounded SQL Tool	73.31% Execution Accuracy	General institutional QA SQL pipeline

Table 12 places this study's Spider results in context. Configuration B achieved 73.31% execution accuracy without any Spider-specific optimization, compared to dedicated systems such as DAIL-SQL (86.2–86.6%) [22] and PET-SQL (87.6%) [23]. The gap is expected, as the proposed SQL tool is a general institutional QA pipeline rather than a specialized Text-to-SQL system. The relatively low SQL Exact Match (0.1963) compared to execution accuracy (0.7331) reflects the well-known characteristic that structurally different SQL queries can yield identical execution results—the more practically relevant criterion in institutional settings. The 82.1% figure for the GPT-3.5/GPT-4 study [21] reflects the best

reported result under fine-tuned GPT-3.5 with GPT-4-based error correction on the Spider development set.

5.2. Query-Type and Error Analysis

SQL tool invocation showed the largest effect on numeric queries, improving accuracy from 43.06% (A) to 88.89% (B and C), demonstrating that document retrieval alone is structurally insufficient for numerical lookups and aggregation. For hybrid queries, the decrease from B (85.42%) to C (71.88%), shown by an exact McNemar test to be statistically non-significant ($p = 0.07$), reflects a deliberate trade-off: the guardrail layer conservatively restricts responses when document evidence and SQL results are inconsistent or when permission scope is ambiguous, prioritizing security over completeness.

Table 13 presents representative failure cases. The remaining failure in Configuration C was a logging schema implementation-level issue rather than an incorrect security decision, confirming that the framework's security design operated as intended. More broadly, failures in the proposed framework occur at four pipeline stages. First, routing errors arise when query type is ambiguous, particularly for hybrid questions requiring both retrieval and SQL. Second, SQL generation errors result from incorrect schema references or improper JOINs, though their impact is partially mitigated by the fact that a structurally different SQL can still yield correct execution results. Third, evidence combination errors occur when the guardrail layer restricts responses due to insufficient or inconsistent evidence across document and SQL sources—desirable from a security standpoint but at the cost of response completeness. Fourth, incorrect security decisions in configurations without guardrails (A0, A1, B) led to high TIL and zero refusal/masking correctness, underscoring that permission-based response control must be an explicit architectural layer, not an emergent property of retrieval or SQL accuracy.

Table 13. Representative failure cases observed in the in-house benchmark.

Config.	Query Type	Error Category	Cause Summary	Implication
A0, A1, B	restricted/adversarial	Security/guardrail failure	No refusal/masking performed; tenant boundary not enforced	Tool-calling alone cannot satisfy security requirements
B	role_restricted	Security/guardrail failure	Responded to restricted query without guardrail layer	Functional accuracy and security control must be designed separately
C	hybrid	SQL generation/execution error	Execution failure due to logging schema constraint	Implementation-level robustness issue, not a reasoning error

To assess the reliability of these differences, we computed exact McNemar tests on paired per-question outcomes and Wilson 95% confidence intervals for each accuracy metric. The improvement from Configuration A to Configuration B was highly significant for both overall accuracy and numeric queries (McNemar exact $p < 0.0001$), confirming that schema-grounded SQL tool-calling, rather than document retrieval, drives the numeric gain. By contrast, the differences between Configurations B and C were not statistically significant for overall accuracy ($p = 0.15$) or hybrid queries ($p = 0.07$), indicating that the accuracy reduction from the guardrail layer reflects a small number of conservative refusals rather than a meaningful loss of utility.

6. Conclusions

This paper proposes and evaluates a tool-calling LLM framework for secure institutional QA in multi-tenant environments. By integrating document retrieval, SQL/BI querying, access control, and guardrails within a unified pipeline, the framework addresses the composite challenge of handling both structured and unstructured data while enforcing tenant isolation and security policy. Progressive addition of SQL tool-calling (Configuration B) raised overall accuracy from 75.25% to 92.65%, while the security and guardrail layer (Configuration C) maintained a comparable 89.46% overall accuracy and simultaneously achieved TIL 0.00%, refusal correctness of 100.00%, and security accuracy of 100.00%. On the Spider benchmark, Configuration B achieved 73.31% execution accuracy without benchmark-specific optimization, providing evidence of the SQL pipeline's executability and portability in an external cross-domain setting. Together, these results demonstrate that SQL tool-calling and security guardrails are both essential—neither alone is sufficient for reliable multi-tenant institutional QA.

6.1. Limitations

The in-house benchmark is limited to a single institution (DLS, Catholic University of Korea), constraining generalizability and precluding full public release of the dataset. The Spider experiments did not apply Text-to-SQL-specific techniques such as few-shot example selection or execution-guided decoding, so the results reflect SQL pipeline executability rather than state-of-the-art performance. Response latency, API cost, and concurrent user handling were not analyzed, and the accuracy–security trade-off observed for hybrid queries warrants further study. In addition, because human-evaluation disagreements were resolved through discussion to consensus rather than preserved as independent ratings, a formal inter-rater reliability statistic could not be computed for the current evaluator panel.

In terms of computational overhead, the additional security and tool-calling stages introduce a bounded, largely deterministic cost on top of the dominant LLM-inference and retrieval latency: query classification is a single lightweight LLM call, SQL validation is a syntactic allowlist/parse check, and guardrail masking/refusal is rule-based string processing. The main variable cost is the optional second LLM call for SQL generation and hybrid composition. A full empirical characterization of latency, API cost, and concurrent-user throughput is left to future work.

6.2. Future Work

Seven directions are proposed for future work: (1) expanding the in-house benchmark to diverse institutional domains (HR, accounting, R&D) and developing independently constructed, anonymized, or synthetic benchmark variants to support third-party evaluation and improved reproducibility; (2) adding HybridQA [17] and FinQA [24] to evaluate document–table hybrid and numerical reasoning capabilities; (3) adapting prompt representation, example selection, and consistency-checking techniques from DAIL-SQL [22] and PET-SQL [23] to improve SQL generation accuracy on both in-house and public benchmarks; (4) expanding the adversarial query set to include indirect prompt injection, multi-turn attacks, schema-inference attempts, role escalation, and mixed authorized/unauthorized queries, while adding latency, cost, and observability analyses for operational applicability; (5) recruiting a larger, independently scored evaluator panel to quantify inter-rater reliability for the human-evaluation protocol; (6) evaluating the robustness of the reported results across alternative LLMs, embedding models, and retrieval hyperparameters (chunk size, top-k, similarity threshold) to establish whether the observed gains generalize beyond the present single-model configuration; and (7) conducting head-to-head comparisons with

representative agentic RAG, SQL-RAG, Text-to-SQL, hybrid table-text QA, and enterprise-grade LLM security frameworks.

Author Contributions: Conceptualization, S.-E.K. and Z.W.G.; methodology, S.-E.K.; software, S.-E.K.; validation, S.-E.K. and Z.W.G.; formal analysis, S.-E.K.; investigation, S.-E.K.; data curation, S.-E.K.; writing—original draft preparation, S.-E.K.; writing—review and editing, S.-E.K. and Z.W.G.; supervision, Z.W.G.; funding acquisition, Z.W.G. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the Korea Institute of Energy Technology Evaluation and Planning (KETEP) and the Ministry of Trade, Industry & Energy, Republic of Korea (RS-2024-00441420; RS-2024-00442817).

Institutional Review Board Statement: Not applicable. This study used only publicly available web materials and did not involve direct recruitment of human participants, intervention, surveys, interviews, experiments, access to login-restricted systems, or collection of private or identifiable personal information by the authors.

Informed Consent Statement: Not applicable. The three evaluators who scored system outputs served in a rating/annotation capacity rather than as research subjects; no personal, identifying, or sensitive information about the evaluators was collected.

Data Availability Statement: The evaluation artifacts and processed benchmark files used in this study—including benchmark question sets, human-evaluation exports, and Spider re-evaluation summaries—are publicly available at https://github.com/friendlywhales/MTF-Tool-Calling-LLMs (accessed on 19 July 2026). The publicly available institutional web materials from which the in-house benchmark was derived can be accessed from the official website of the Department of Liberal Studies at the Catholic University of Korea: https://liberal.catholic.ac.kr/liberal/index.do (accessed on 19 July 2026). The internally constructed database tables and document corpus derived from these publicly available web materials have not been released. No private student records or personally identifiable records were used in this study. The evaluation code itself has not been released. The public Spider dev set is available from its original source.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Lewis, P.; Perez, E.; Piktus, A.; Petroni, F.; Karpukhin, V.; Goyal, N.; Kuettler, H.; Lewis, M.; Yih, W.T.; Rocktaeschel, T.; et al. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. *Adv. Neural Inf. Process. Syst.* **2020**, *33*, 9459–9474.
2. Gao, Y.; Xiong, Y.; Gao, X.; Jia, K.; Pan, J.; Bi, Y.; Dai, Y.; Sun, J.; Guo, M.; Wang, H.; et al. Retrieval-Augmented Generation for Large Language Models: A Survey. *arXiv* **2024**, arXiv:2312.10997. [CrossRef]
3. Yu, T.; Zhang, R.; Yang, K.; Yasunaga, M.; Wang, D.; Li, Z.; Ma, J.; Li, I.; Yao, Q.; Roman, S.; et al. Spider: A Large-Scale Human-Labeled Dataset for Complex and Cross-Domain Semantic Parsing and Text-to-SQL Task. In Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing (EMNLP), Brussels, Belgium, 31 October–4 November 2018; pp. 3911–3921.
4. Karpukhin, V.; Oguz, B.; Min, S.; Lewis, P.; Wu, L.; Edunov, S.; Chen, D.; Yih, W.T. Dense Passage Retrieval for Open-Domain Question Answering. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), Online, 16–20 November 2020; pp. 6769–6781. [CrossRef]
5. Khatri, V.; Brown, C.V. Designing Data Governance. *Commun. ACM* **2010**, *53*, 148–152. [CrossRef]
6. Yao, Y.; Duan, J.; Xu, K.; Cai, Y.; Sun, Z.; Zhang, Y. A Survey on Large Language Model (LLM) Security and Privacy: The Good, the Bad, and the Ugly. *High-Confid. Comput.* **2024**, *4*, 100211. [CrossRef]
7. Kim, S.E.; Geem, Z.W. Case Study: Tool-Calling RAG with Private LLM in Multi-Tenant Enterprise Environment. *J. Korean Inst. Intell. Syst.* **2026**, *36*, 42–48. [CrossRef]
8. Ni, B.; Liu, Z.; Wang, L.; Lei, Y.; Zhao, Y.; Cheng, X.; Zeng, Q.; Dong, L.; Xia, Y.; Kenthapadi, K.; et al. Towards Trustworthy Retrieval Augmented Generation for Large Language Models: A Survey. *arXiv* **2025**, arXiv:2502.06872. [CrossRef]
9. Yao, S.; Zhao, J.; Yu, D.; Du, N.; Shafran, I.; Narasimhan, K.; Cao, Y. ReAct: Synergizing Reasoning and Acting in Language Models. In Proceedings of the International Conference on Learning Representations (ICLR), Kigali, Rwanda, 1–5 May 2023.

10. Qin, Y.; Liang, S.; Ye, Y.; Zhu, K.; Yan, L.; Lu, Y.; Lin, Y.; Cong, X.; Tang, X.; Qian, B.; et al. ToolLLM: Facilitating Large Language Models to Master 16000+ Real-World APIs. *arXiv* **2023**, arXiv:2307.16789.
11. Ali, Z.; Huang, Y.; Khan, A.; Qi, G.; Zhang, Y.; Feng, J.; Deng, C.; Kefalas, P. Pythia-RAG: Retrieval-Augmented Generation over a Unified Multimodal Knowledge Graph for Enhanced QA. *Knowl.-Based Syst.* **2026**, *335*, 115200. [CrossRef]
12. Khan, A.; Ali, Z.; Irfanullah, A.A.; Kefalas, P. Talk2Doc: A Patient Q&A System Using Retrieval-Augmented Generation with Weighted Knowledge Graphs and LLMs. In Proceedings of the 2025 International Conference on Intelligent Computing (ICIC 2025), Ningbo, China, 26–29 July 2025. Available online: http://poster-openaccess.com/files/ICIC2025/3510.pdf (accessed on 2 July 2026).
13. Zhong, V.; Xiong, C.; Socher, R. Seq2SQL: Generating Structured Queries from Natural Language Using Reinforcement Learning. *arXiv* **2017**, arXiv:1709.00103. [CrossRef]
14. Kim, H.; So, B.-H.; Han, W.-S.; Lee, H. Natural Language to SQL: Where Are We Today? *Proc. VLDB Endow.* **2020**, *13*, 1737–1750. [CrossRef]
15. Fu, H.; Liu, C.; Wu, B.; Li, F.; Tan, J.; Sun, J. CatSQL: Towards Real World Natural Language to SQL Applications. *Proc. VLDB Endow.* **2023**, *16*, 1534–1547. [CrossRef]
16. Pourreza, M. Text-to-SQL Systems in the Era of Advanced Large Language Models. Master's Thesis, University of Alberta, Edmonton, AB, Canada, 2024.
17. Chen, W.; Zha, H.; Chen, Z.; Xiong, W.; Wang, H.; Wang, W.Y. HybridQA: A Dataset of Multi-Hop Question Answering over Tabular and Textual Data. In Proceedings of the Association for Computational Linguistics: EMNLP 2020, Online, 16–20 November 2020; pp. 1026–1036.
18. Chen, W.; Chang, M.W.; Schlinger, E.; Wang, W.Y.; Cohen, W.W. Open Question Answering over Tables and Text. In Proceedings of the 9th International Conference on Learning Representations (ICLR), Virtual, 3–7 May 2021.
19. Wang, D.; Dou, L.; Che, W. A Survey on Table-and-Text HybridQA: Concepts, Methods, Challenges and Future Directions. *arXiv* **2022**, arXiv:2212.13465. [CrossRef]
20. Glenn, P.; Bhatt, U.; Koreeda, Y.; Jain, A.; Zhao, T.Z.; Potts, C.; Manning, C.D.; Zou, J. BlendSQL: A Scalable Dialect for Unifying Hybrid Question Answering in Relational Algebra. In Proceedings of the Association for Computational Linguistics: ACL 2024, Bangkok, Thailand, 11–16 August 2024; pp. 7203–7220.
21. Roberson, R.; Kaki, G.; Trivedi, A. Analyzing the Effectiveness of Large Language Models on Text-to-SQL Synthesis. *arXiv* **2024**, arXiv:2401.12379.
22. Gao, D.; Wang, H.; Li, Y.; Sun, X.; Qian, Y.; Ding, B.; Zhou, J. Text-to-SQL Empowered by Large Language Models: A Benchmark Evaluation. *Proc. VLDB Endow.* **2024**, *17*, 1132–1145. [CrossRef]
23. Li, Z.; Wang, X.; Zhao, J.; Yang, S.; Du, G.; Hu, X.; Zhang, B.; Ye, Y.; Li, Z.; Zhao, R.; et al. PET-SQL: A Prompt-Enhanced Two-Round Refinement of Text-to-SQL with Cross-Consistency. *arXiv* **2024**, arXiv:2403.09732.
24. Chen, Z.; Chen, W.; Smiley, C.; Shah, S.; Borova, I.; Langdon, D.; Moussa, R.; Beane, M.; Huang, T.H.; Routledge, B.; et al. FinQA: A Dataset of Numerical Reasoning over Financial Data. In Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing (EMNLP), Punta Cana, Dominican Republic, 7–11 November 2021; pp. 3697–3711.