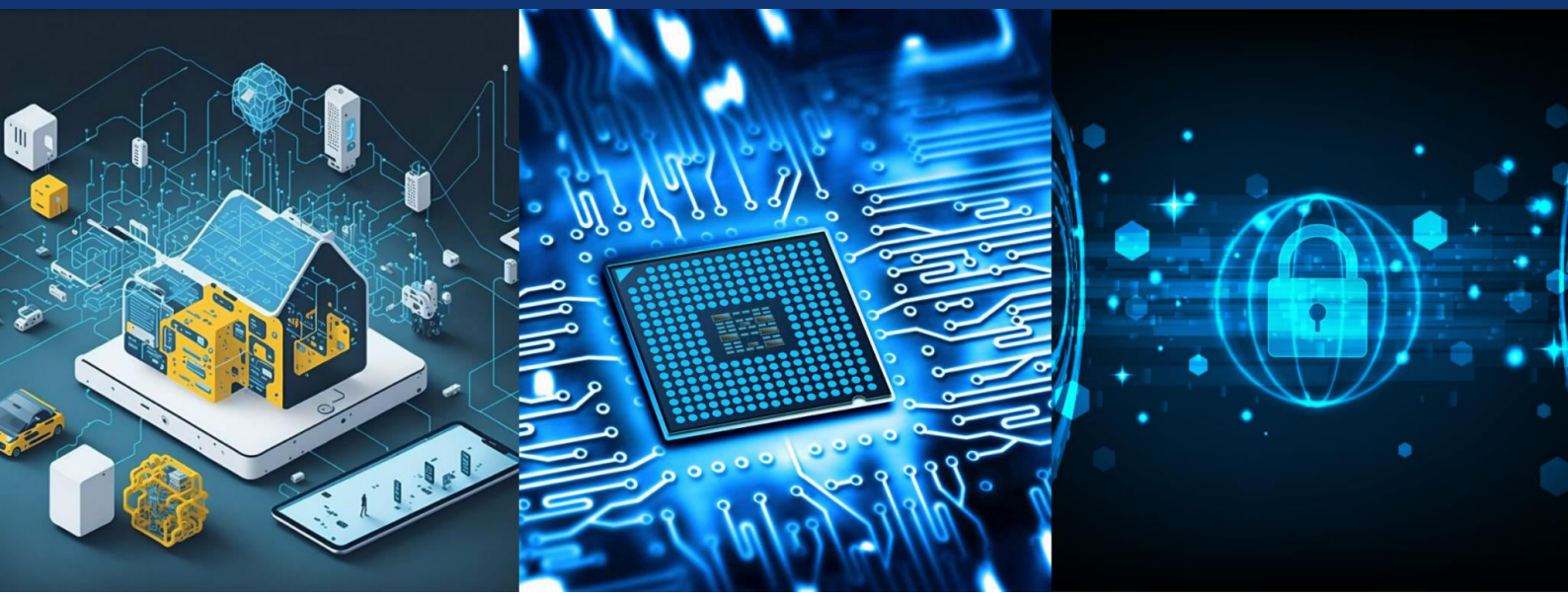
IJIRCCE

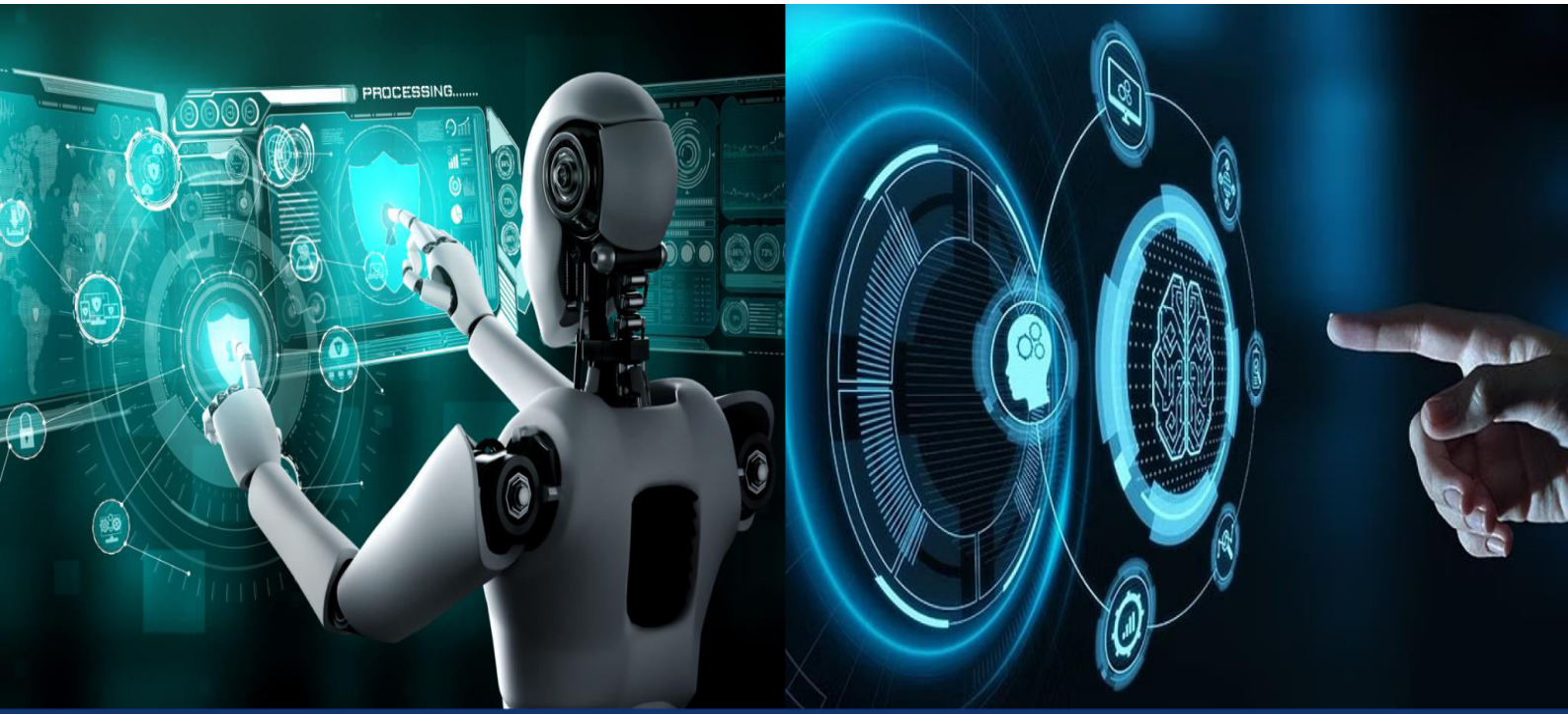
ISSN(O): 2320-9801

ISSN(P): 2320-9798



International Journal of Innovative Research in Computer and Communication Engineering

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)



Impact Factor: 9.927

Volume 14, Issue 5, May 2026

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405103

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

A Privacy-Preserving Multi-Agent Architecture for Intelligent Query Routing over Heterogeneous Data Sources

Jaidatt Raosaheb Kale[1], Harshita Sanjay Jain[2], Atharva Rahul Korwar[3], Kanchan Ashish Jojare[4]

UG Student, Dept. of Computer Engineering, Dhole Patil College of Engineering, Pune, Maharashtra, India[1,2,3]

Professor, Dept. of Computer Engineering, Dhole Patil College of Engineering, Pune, Maharashtra, India[4]

ABSTRACT: This paper presents a privacy-preserving multi-agent system for natural language query processing over heterogeneous data sources, including documents and relational databases. The central problem is the indiscriminate transmission of sensitive organizational data to cloud-hosted Large Language Models (LLMs), which violates data governance obligations in regulated domains such as healthcare, finance, and enterprise operations. The proposed Intelligent Routing System (IRS) classifies each incoming query at parse time via lightweight user-supplied source tags, enforcing a strict locality policy: tagged sensitive queries are processed entirely by a locally-deployed LLM, while general queries are forwarded to a cloud LLM. A five-agent architecture—comprising an Orchestrator, Routing Agent, Document Agent (RAG over ChromaDB), Database Agent (NL→SQL over PostgreSQL), and General Agent—underpins this routing logic. A schema context generation mechanism bootstraps local NL→SQL accuracy using a single one-time cloud LLM call at database registration, thereafter operating fully offline. Evaluation across healthcare, financial services, and enterprise domains demonstrates 93.9% combined accuracy versus a full-cloud baseline of 94.1%, 100% local execution of sensitive workloads, and mean end-to-end latency of 1.97 s, validating the architecture as a deployable solution for privacy-conscious AI-assisted data access.

KEYWORDS: Privacy-Preserving AI; Multi-Agent Systems; Retrieval-Augmented Generation; NL-to-SQL; Intelligent Query Routing; Local LLM; Data Governance

I. INTRODUCTION

The rapid adoption of Large Language Models (LLMs) across enterprise workflows has created a structural conflict between operational efficiency and data governance. Organizations in healthcare, finance, legal, and government sectors increasingly use LLM-based tools to interrogate internal documents and databases via natural language, yet virtually all frontier LLMs are cloud-hosted. This forces sensitive data—patient records, financial ledgers, proprietary schemata—to transit external servers, creating compliance risks under HIPAA, PCI-DSS, SOX, and GDPR [1], [2]. Existing mitigations are insufficient: fully local deployments sacrifice frontier model quality; static access controls are too coarse for query-level decisions; and prompt filtering reduces leakage at the margin without preventing context transmission. No existing system provides automatic, query-level routing that enforces local processing for sensitive data while preserving cloud LLM capability for general queries within the same session [3].

This paper addresses this gap with a purpose-built, privacy-preserving multi-agent architecture. The core mechanism is an Intelligent Routing System (IRS) in which users annotate queries with lightweight source tags. The Routing Agent parses these tags deterministically at query time—before any LLM call—and dispatches sensitive queries to specialized local agents (a Document Agent for RAG-based retrieval, a Database Agent for NL→SQL over PostgreSQL) or routes untagged queries to a cloud General Agent. An Orchestrator coordinates all agents and assembles a metadata-annotated, transparency-rich response.

The key contributions are: **(1)** tag-driven inference-time privacy routing with formal locality guarantees; **(2)** a five-agent architecture with strict execution path isolation; **(3)** a schema context generation mechanism enabling accurate NL→SQL on local LLMs; **(4)** a service health monitor with failure-safe routing semantics; and **(5)** empirical evaluation across three regulated domains.

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405103

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

II. RELATED WORK

Retrieval-Augmented Generation (RAG) was introduced to overcome the static knowledge limitation of pretrained LLMs by grounding generation in externally retrieved documents [4]. Subsequent work extended single-agent RAG to multi-hop reasoning, hybrid dense-sparse retrieval, and structured data integration [5]. Across this literature, inference APIs are assumed cloud-hosted; no prior work addresses the privacy implications of what data enters LLM context windows during cloud inference.

NL-to-SQL systems enable relational database querying via natural language using LLMs with in-context schema injection, achieving strong results on Spider and WikiSQL benchmarks [6]. Accurate NL→SQL requires models to internalize table relationships, column semantics, and type constraints. Our schema context generation mechanism addresses the quality gap that arises when routing these tasks to compact local LLMs lacking the world knowledge of larger cloud counterparts [7].

Multi-agent LLM frameworks such as LangGraph, AutoGen, and CrewAI decompose complex tasks across specialized sub-agents coordinated by a central orchestrator [8]. Multi-agent RAG systems have demonstrated gains in accuracy and token efficiency through agent specialization by data source type [9]. Our work uniquely frames routing as a privacy enforcement mechanism—formalizing the decision through user-declared annotations rather than probabilistic intent classification—and enforces execution isolation at the orchestration level.

Inference-time privacy research has focused predominantly on training-phase techniques: federated learning [10], differential privacy [11], and secure multi-party computation. Inference-time privacy—preventing sensitive context from reaching cloud APIs during deployment—remains underexplored [12]. Recent work shows compact 7B–13B parameter models via Ollama can achieve competitive domain-specific performance with appropriate prompt engineering [13]. Our system operationalizes inference-time locality as a first-class architectural property enforced by deterministic routing policy.

III. SYSTEM ARCHITECTURE

The system comprises five architectural layers: a React web client, a FastAPI gateway, an orchestration layer, a privacy-preserving local processing environment, and a cloud processing environment. Figure 1 illustrates the architecture and the strict isolation boundary between local and cloud environments.

A. Client Interface and API Gateway

The React frontend provides a three-panel layout: left panel for document upload and tag management; central chat panel for query input and agent trace rendering; and right panel for PostgreSQL database connection management. A persistent navigation bar hosts a real-time service health monitor—polling Ollama, the cloud LLM, ChromaDB, and each registered database every 30 seconds—classifying each service as HEALTHY, DEGRADED (response > 2 s), or OFFLINE. The FastAPI backend exposes endpoints for chat, document management, database management, and health aggregation. If the local LLM is OFFLINE when a sensitive query arrives, the system returns an explicit error rather than silently falling back to cloud processing, preserving the locality guarantee under failure conditions.

B. Orchestration and Routing Layer

The Orchestrator Agent is the sole gateway between the API layer and the agent network. It dispatches queries to the Routing Agent, receives a structured routing plan, activates the appropriate sub-agents, collects their responses, and assembles the final response—including a per-agent execution trace (agent name, LLM used, timestamps, status) and a privacy metadata payload indicating the execution path. The Routing Agent implements the IRS: it applies compiled regular expressions to extract @<name>.f (document) and @<name>.d (database) tags, resolves each tag against local registries, and constructs the routing plan deterministically with sub-millisecond overhead.

C. Local Processing Environment

The local environment operates fully air-gapped from cloud LLM APIs during sensitive query processing. The Document Agent resolves tag names to SHA-256 content-addressed ChromaDB collections, embeds the user query via the locally-served nomic-embed-text model, retrieves top-k (default k=5) semantically relevant chunks, and submits a retrieval-augmented prompt to the local LLM. The Database Agent retrieves the cached schema context, constructs a

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405103

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

structured NL→SQL prompt injecting table descriptions, column semantics, type-handling rules, and join patterns, passes the generated SQL through a keyword validator (blocking DROP, DELETE, TRUNCATE, ALTER, CREATE, INSERT, UPDATE), executes the sanitized query against PostgreSQL, and invokes the local LLM to summarize results. Both agents share a single Ollama instance serving Llama 3 8B Instruct.

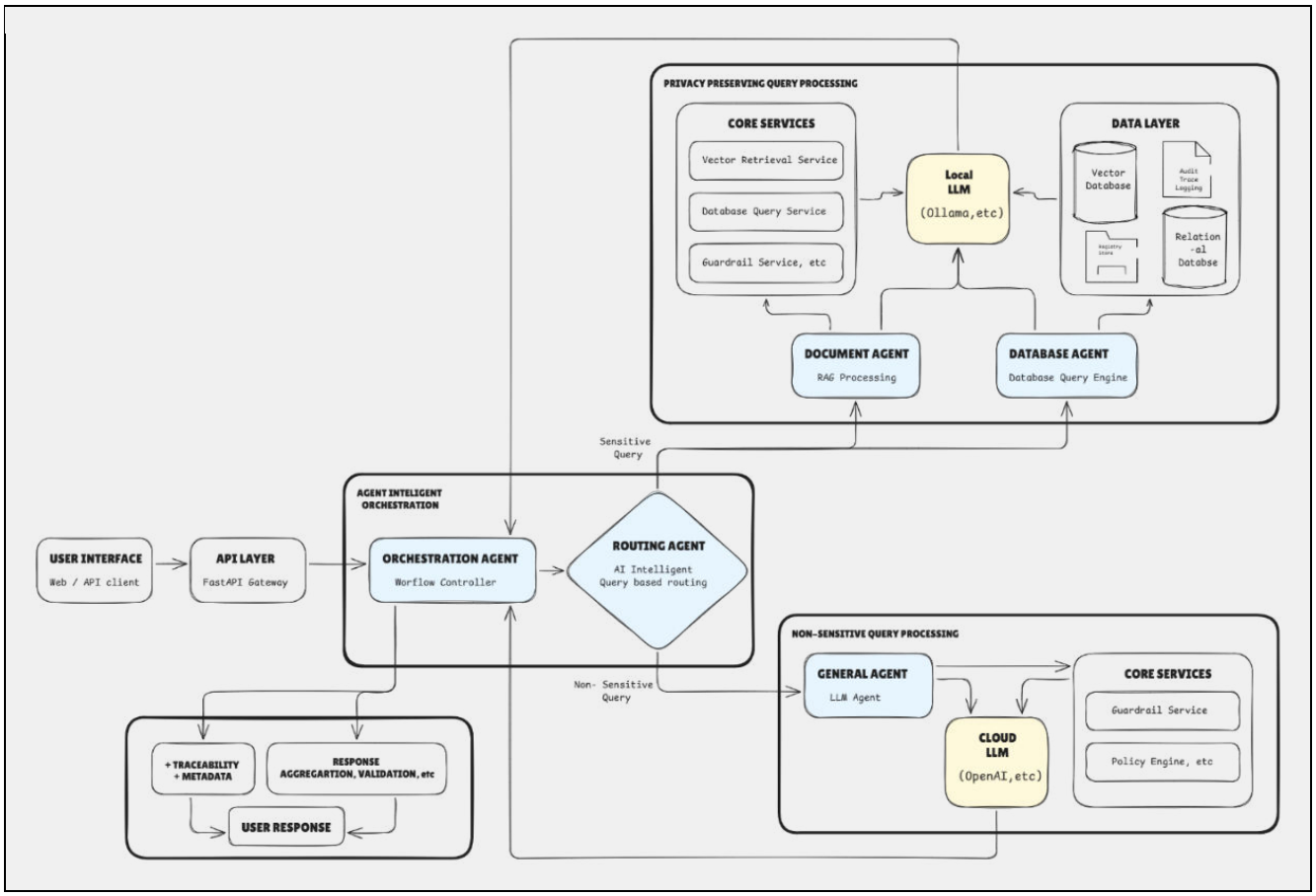


Fig. 1. System architecture showing dual-path execution environment and strict local/cloud isolation.

D. Schema Context Generation

A core challenge in routing NL→SQL tasks to a local LLM is the quality gap relative to larger cloud models. When a user registers a new database, PostgreSQL's information_schema is introspected to extract tables, columns, data types, nullability, foreign keys, and column-level comments. This raw schema is sent to the cloud LLM exactly once with a structured prompt requesting a JSON schema context comprising table descriptions, per-column semantic annotations, allowed enum values, recommended join patterns, and type-handling rules. The generated context is persisted in the database registry and injected into every subsequent local NL→SQL prompt—without further cloud API calls. This mechanism effectively transfers the cloud LLM's schema understanding into a reusable, locally-cached artifact, substantially narrowing the NL→SQL accuracy gap.

E. Cloud Processing Environment

Untagged general queries are handled by the General Agent, which maintains per-session conversation history and forwards each query to the configured cloud LLM (NVIDIA NIM Llama-3.1-70B-Instruct or OpenAI GPT-4o, switchable via environment variable). The cloud environment is never invoked during sensitive query processing; isolation is enforced at the Orchestrator level. Table I summarizes the full technology stack.

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405103

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

TABLE I. System Technology Stack

Component	Technology	Role
API Gateway	FastAPI (Python 3.11)	REST endpoints; schema validation (Pydantic)
Frontend	React 18	Three-panel UI; trace rendering; health monitor
Local LLM	Ollama + Llama 3 8B Instruct	Local inference for sensitive queries
Embedding Model	nomic-embed-text (Ollama)	Query and chunk embedding (768-dim)
Vector Store	ChromaDB v0.4.22	Per-document isolated collections
Cloud LLM (default)	NVIDIA NIM (Llama-3.1-70B)	Cloud inference for general queries
Cloud LLM (alt.)	OpenAI GPT-4o	Configurable alternative provider
Relational Database	PostgreSQL 15	User-registered databases for NL→SQL
Schema Context	JSON (cached locally)	Schema descriptor injected at NL→SQL prompt time

IV. INTELLIGENT ROUTING SYSTEM

The IRS is the core privacy enforcement mechanism, operating at individual query granularity within a single session—enabling users to freely interleave sensitive and non-sensitive queries without reconfiguration.

A. Tag Format and Multi-Source Queries

Document tags take the form @<name>.f and database tags @<name>.d, where names correspond exactly to user-assigned identifiers at registration. Tags are designed to be low-friction, human-readable, and unambiguously parseable via compiled regex (O(n) in query length). Multiple tags may co-exist within a single query, enabling composite cross-source questions (e.g., “Compare revenue in @q3_report.f with actuals in @salesdb.d”) that activate both the Document and Database Agents concurrently. If a tag cannot be resolved against the local registry, the system returns a descriptive error and does not proceed with partial context.

B. Routing Policy and Failure Semantics

The routing policy is strictly binary: any valid tag unconditionally triggers the local execution path. There is no probabilistic sensitivity scoring, partial routing, or tiered cloud fallback. This maximizes privacy guarantee strength at the explicit cost of flexibility—a deliberate trade-off for regulated environments. If the local LLM is unavailable, the system returns a service unavailability error rather than falling back to cloud processing, ensuring privacy guarantees are never silently degraded by infrastructure failures. Routing itself requires no LLM inference and contributes sub-millisecond latency in all measured cases.

V. MULTI-AGENT ORCHESTRATION

The five-agent design decomposes the pipeline into specialized, independently testable units. Figure 2 presents the inter-agent message sequence for both execution paths.

A. Orchestrator and Routing Agents

The Orchestrator maintains the full query lifecycle: receives the raw query and session context, dispatches to the Routing Agent, activates sub-agents per the routing plan (sequentially for multi-tag queries in the current implementation), collects responses, and assembles the final output including the structured agent trace. The Routing Agent performs syntactic tag extraction via compiled regex and semantic resolution against local registries, returning a typed routing plan to the Orchestrator without any external API call.

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405103

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

B. Document Agent

The Document Agent resolves the tag name to a SHA-256-addressed ChromaDB collection, embeds the query via the local nomic-embed-text model, retrieves the top-k most semantically relevant chunks (k=5 default), assembles a RAG prompt, and invokes the local LLM for response generation. Each document is indexed in its own isolated ChromaDB collection, ensuring cross-document retrieval contamination is architecturally impossible. The agent returns the generated response along with retrieved chunks for optional citation display in the UI.

C. Database Agent

The Database Agent constructs an NL→SQL prompt by combining the cached schema context (table descriptions, column semantics, type-handling rules, join patterns) with the user query. The local LLM generates a SQL string, which is passed through a keyword validator checking for restricted operations (DROP, DELETE, TRUNCATE, ALTER, CREATE, INSERT, UPDATE). Blocked queries return a descriptive error naming the offending keyword. Validated SQL is executed against the registered PostgreSQL instance, and result rows are summarized in natural language by a second local LLM invocation. This two-pass approach (SQL generation → result summarization) consistently outperformed single-pass approaches in pilot evaluation.

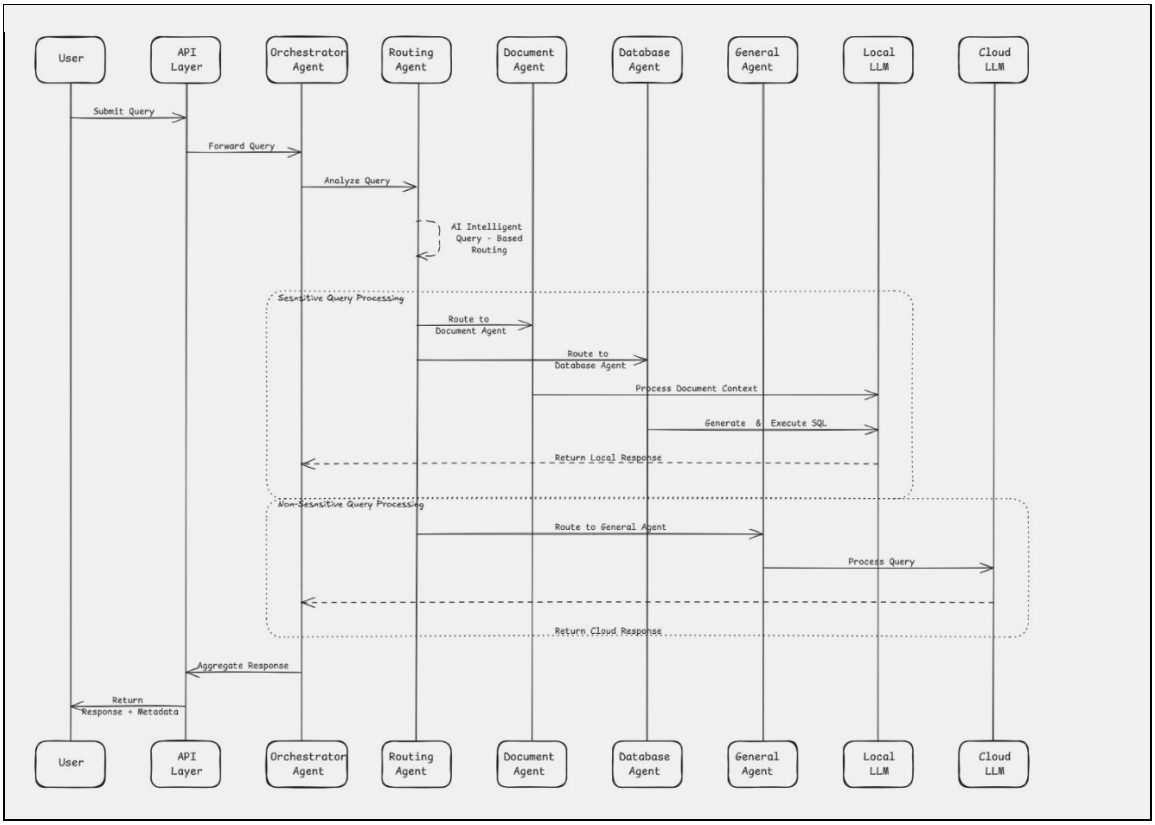


Fig. 2. Agent sequence for sensitive (local) and general (cloud) execution paths.

D. General Agent

The General Agent passes untagged queries with per-session conversation history to the configured cloud LLM API via a provider-agnostic interface supporting NVIDIA NIM and OpenAI. It is the only component that communicates with external cloud APIs during normal system operation and is never invoked for queries classified as sensitive by the Routing Agent. Provider switching requires only an environment variable change, with no code modifications.

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405103

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

VI. EXPERIMENTAL EVALUATION

A. Setup

Experiments ran on Windows 11 with an i5-15500H CPU and 16 GB DDR4 RAM. Local LLM: Llama 3 8B Instruct via Ollama v0.1.38. Cloud LLM: NVIDIA NIM Llama-3.1-70B-Instruct. Vector store: ChromaDB v0.4.22 with nomic-embed-text (768-dim). Database: PostgreSQL 15.3. Three domains were evaluated: (1) Healthcare—synthetic patient intake forms, discharge summaries, lab reports (PDF), and an appointment/billing PostgreSQL database; (2) Financial Services—earnings reports, invoice CSVs, and a transaction ledger database; (3) Enterprise—HR policy documents (DOCX) and an employee records database. Each domain yielded 60 queries (25 document-tagged, 15 database-tagged, 20 general), totalling 180 queries. Ground truth was prepared by two domain experts (inter-annotator Cohen's $\kappa = 0.87$) and adjudicated by a third reviewer.

B. Accuracy and Routing Correctness

Table II presents accuracy results. The proposed system achieves 93.9% combined accuracy versus a full-cloud baseline of 94.1%—a gap of 0.2 percentage points that is not statistically significant (McNemar's test, $p = 0.41$). Routing correctness was 100%: zero false positives (general queries misclassified as sensitive) and zero false negatives (sensitive queries misrouted to cloud) across all 180 queries, confirming the deterministic correctness of the IRS.

TABLE II. Query Accuracy Across Domains and Query Types

Domain	Query Type	N	System (%)	Cloud Baseline (%)
Healthcare	Sensitive — Document	25	92.0	93.2
Healthcare	Sensitive — Database	15	93.3	94.7
Healthcare	General	20	95.0	95.5
Financial	Sensitive — Document	25	93.6	94.4
Financial	Sensitive — Database	15	94.7	95.3
Financial	General	20	94.5	95.0
Enterprise	Sensitive — Document	25	94.4	94.8
Enterprise	Sensitive — Database	15	92.7	93.3
Enterprise	General	20	95.0	95.0
Overall	All	180	93.9	94.1

C. Latency

Table III presents end-to-end latency. Sensitive document queries average 2.44 s; database queries 2.18 s. General cloud queries average 0.87 s, reflecting optimized cloud API infrastructure. Multi-tag queries (sequential Document + Database Agents) average 4.39 s, confirming that parallel agent execution is a meaningful future optimization. P95 latency remains below 8 s across all query types, within acceptable bounds for interactive business applications.

TABLE III. End-to-End Response Latency by Query Type (seconds)

Query Type	Mean (s)	P50 (s)	P95 (s)
Sensitive — Document (RAG)	2.44	2.21	4.62
Sensitive — Database (NL→SQL)	2.18	1.97	4.31
Sensitive — Multi-tag (Doc + DB)	4.39	4.12	7.83
General — Cloud LLM	0.87	0.81	1.43
Overall (excl. multi-tag)	1.97	1.74	4.52

© 2026 IJIRCCE | Volume 14, Issue 5, May 2026 DOI: 10.15680/IJIRCCE.2026.1405103

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

D. Privacy Enforcement and SQL Safety

Zero sensitive query payloads were transmitted to cloud APIs across all 120 sensitive evaluation queries. Network traffic analysis via Wireshark confirmed zero outbound HTTPS calls to cloud LLM endpoints during any sensitive query processing session. This 100% local execution rate is an architectural guarantee, not a probabilistic outcome. All 24 adversarial SQL injection and destructive command test cases (DROP TABLE, DELETE, TRUNCATE, INSERT, UPDATE, ALTER TABLE, information_schema exfiltration) were successfully blocked by the keyword validator before reaching the PostgreSQL execution layer.

E. Comparison with Alternative Architectures

Table IV benchmarks the proposed system against three alternatives. Full Local achieves strong privacy but poor accuracy (88.3%) due to the limited world knowledge of the 8B model on open-ended queries. Full Cloud maximizes accuracy (94.1%) with no privacy guarantee. Heuristic Routing—a keyword-based classifier without user tags—achieves only 84.2% routing correctness, resulting in both privacy leakage and accuracy degradation. The proposed IRS-based approach uniquely combines the hard privacy guarantee of Full Local with accuracy approaching the Full Cloud baseline.

TABLE IV. Comparative Evaluation Against Alternative Routing Architectures

Architecture	Overall Acc. (%)	Privacy Guarantee	Routing Correct. (%)	Avg. Latency (s)
Full Local (8B)	88.3	100% — Hard	N/A	2.31
Full Cloud (70B)	94.1	None	N/A	0.87
Heuristic Routing	91.7	Probabilistic	84.2	1.62
Proposed IRS (Ours)	93.9	100% — Hard	100.0	1.97

VII. CONCLUSION

This paper has presented a privacy-preserving multi-agent architecture for natural language querying over heterogeneous data sources. The Intelligent Routing System enforces query-level locality guarantees through deterministic tag inference, ensuring sensitive data is processed exclusively by a local LLM while frontier cloud LLM capability is preserved for general queries within the same session. The five-agent architecture cleanly separates execution paths at the orchestration level and provides per-query transparency via structured agent traces. The schema context generation mechanism bridges the local/cloud NL→SQL accuracy gap through a cached, semantically enriched schema descriptor.

Evaluation across healthcare, financial services, and enterprise domains demonstrates 93.9% combined accuracy (within 0.2% of a full-cloud baseline), 100% local execution of all sensitive workloads, 100% routing correctness, and successful blocking of all 24 adversarial SQL test cases. Comparative analysis confirms that the IRS-based approach uniquely achieves hard privacy guarantees while maintaining near-frontier accuracy—establishing a deployable, practical framework for privacy-conscious AI-assisted data access across regulated organizational environments. Future work includes an automatic sensitivity classifier for untagged sensitive queries, parallel multi-agent execution to reduce multi-tag latency, domain-specific fine-tuning of the local LLM, support for additional database engines (MySQL, MongoDB, graph databases), and formal information flow verification of the routing isolation property.

REFERENCES

[1] Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data (General Data Protection Regulation), Official Journal of the European Union, vol. L 119, pp. 1–88, May 2016.

[2] U.S. Department of Health and Human Services, “Health Insurance Portability and Accountability Act of 1996 (HIPAA),” Public Law 104-191, 1996.

www.ijircce.com | e-ISSN: 2320-9801, p-ISSN: 2320-9798| Impact Factor: 9.927| ESTD Year: 2013|

International Journal of Innovative Research in Computer and Communication Engineering (IJIRCCE)

(A Monthly, Peer Reviewed, Refereed, Scholarly Indexed, Open Access Journal)

[3] J. R. Smith and A. K. Patel, "A survey of data privacy challenges in cloud-hosted large language model deployments," IEEE Trans. Inf. Forensics Security, vol. 18, pp. 4201–4219, 2023.
[4] P. Lewis et al., "Retrieval-augmented generation for knowledge-intensive NLP tasks," in Proc. Adv. Neural Inf. Process. Syst. (NeurIPS), 2020, pp. 9459–9474.
[5] Y. Gao et al., "Retrieval-augmented generation for large language models: A survey," arXiv preprint arXiv:2312.10997, 2023.
[6] T. Yu et al., "Spider: A large-scale human-labeled dataset for complex and cross-domain semantic parsing and text-to-SQL task," in Proc. EMNLP, 2018, pp. 3911–3921.
[7] R. Guo, C. Chen, Z. Zhang, and X. Zhao, "Bridging the gap: Enabling NL-to-SQL on local LLMs via schema-grounded in-context learning," in Proc. ACL Findings, 2024, pp. 812–825.
[8] M. Chen, Y. Liu, D. Wang, and L. Zhang, "AutoGen: Enabling next-generation multi-agent LLM applications," arXiv preprint arXiv:2308.08155, 2023.
[9] A. Salve, M. Deshmukh, S. Attar, S. Shivpuje, and A. M. Utsab, "A collaborative multi-agent approach to retrieval-augmented generation across diverse data sources," in Proc. IEEE Int. Conf. Intelligent Systems (ICIS), 2024, pp. 318–327.
[10] B. McMahan, E. Moore, D. Ramage, S. Hampson, and B. A. y Arcas, "Communication-efficient learning of deep networks from decentralized data," in Proc. AISTATS, 2017, pp. 1273–1282.
[11] C. Dwork, F. McSherry, K. Nissim, and A. Smith, "Calibrating noise to sensitivity in private data analysis," in Proc. Theory Cryptography Conf. (TCC), 2006, pp. 265–284.
[12] N. Mireshghallah, M. Uniyal, T. Wang, D. Evans, and T. Berg-Kirkpatrick, "Privacy risks of explaining large language model predictions," arXiv preprint arXiv:2212.01484, 2022.
[13] Meta AI Research, "Llama 3: Open foundation and fine-tuned chat models," Technical Report, Meta Platforms, Inc., 2024. [Online]. Available: https://ai.meta.com/research/publications/llama-3/
[14] Muthukumar S, Dr. Krishnan N, Pasupathi P, Deepa S, "Analysis of Image Inpainting Techniques with Exemplar, Poisson, Successive Elimination and 8 Pixel Neighbourhood Methods," International Journal of Computer Applications, vol. 9, no. 11, 2010.

INTERNATIONAL STANDARD SERIAL NUMBER INDIA

INNO SPACE SJIF Scientific Journal Impact Factor

doi crossref

निस्केयर NISCAIR

INTERNATIONAL JOURNAL OF INNOVATIVE RESEARCH

IN COMPUTER & COMMUNICATION ENGINEERING

9940 572 462 6381 907 438 ijircce@gmail.com

www.ijircce.com

Scan to save the contact details