Accurate Enterprise AI Analytics:

A Reference Architecture for Anti-Hallucination SQL-RAG Systems on Oracle Cloud Infrastructure

Ameer Fahad Ali Khan

Oracle Corporation

ameerfahadali@gmail.com

December 2025

Keywords: *Retrieval-Augmented Generation, Natural Language to SQL, Large Language Models, Anti-Hallucination, Enterprise AI, Oracle Cloud Infrastructure, Metadata Filtering*

Abstract

Enterprise adoption of Large Language Models (LLMs) for business analytics faces critical challenges: hallucination of non-existent data, inconsistent retrieval from document repositories, and the absence of programmatic metadata filtering in cloud AI services. This paper presents a reference architecture for enterprise AI analytics that integrates natural language SQL generation with Retrieval-Augmented Generation (RAG) on Oracle Cloud Infrastructure. We introduce novel anti-hallucination patterns for agent design, document an undiscovered API format for metadata filtering in OCI GenAI Agents (discovered through reverse engineering), and demonstrate a multi-model orchestration approach that achieves 30% efficiency improvement over traditional analytics platforms. The architecture has been validated in production deployment for a major retail corporation, processing queries against 41.5 million transaction records and 77 corporate documents. Our contributions include: (1) a reusable anti-hallucination constraint framework for SQL-RAG agents, (2) the first documented working implementation of programmatic metadata filtering in OCI GenAI Agent Runtime API, and (3) performance benchmarks comparing AI-driven analytics against Snowflake-based solutions. The system demonstrates 100% metadata filtering accuracy compared to 12.5% without filtering, and sub-second cached response times for complex analytical queries.

1. Introduction

The integration of Large Language Models into enterprise analytics systems represents both an opportunity and a challenge. While LLMs enable natural language interfaces that democratize access to business intelligence, they introduce risks that are unacceptable in enterprise contexts—most notably, the tendency to hallucinate plausible-sounding but entirely fabricated information [1]. This paper addresses the fundamental question: How can organizations deploy AI-powered analytics systems that are both accessible and trustworthy?

We present a comprehensive reference architecture developed and validated through production deployment for Metro Retail, a major Philippine retail corporation with operations spanning 192 stores and processing 41.5 million annual transactions. The system integrates three distinct AI capabilities: natural language to SQL generation, Retrieval-Augmented Generation from corporate documents, and a hybrid mode that combines both approaches for complex analytical queries.

1.1 Problem Statement

Enterprise AI analytics systems face several interconnected challenges:

1. **Hallucination Risk:** LLMs can generate confident responses containing fabricated metrics, non-existent business goals, or misattributed financial data—potentially leading to costly business decisions based on false information.

2. **Retrieval Precision:** RAG systems without metadata filtering return contextually irrelevant documents, reducing answer quality and user trust. Our initial implementation showed only 12.5% document relevance without filtering.

3. **API Limitations:** Cloud AI services often provide capabilities in graphical interfaces that are not exposed through programmatic APIs, limiting enterprise integration possibilities.

4. **Multi-Model Coordination:** Different AI tasks require different model characteristics—SQL generation benefits from chain-of-thought reasoning while text rephrasing requires speed over depth.

1.2 Contributions

This paper makes the following contributions:

- A reusable **anti-hallucination constraint framework** for enterprise AI agents, with specific patterns for financial data accuracy
- The **first documented working implementation** of programmatic metadata filtering in OCI GenAI Agent Runtime API, discovered through console UI reverse engineering
- A **multi-model orchestration architecture** that optimizes for both accuracy and performance across different AI task types
- **Production-validated performance benchmarks** demonstrating 30% efficiency improvement over Snowflake-based analytics

2. Related Work

2.1 Natural Language to SQL

The translation of natural language queries to SQL has been extensively studied, with approaches ranging from rule-based systems [2] to neural sequence-to-sequence models [3]. Recent work has focused on leveraging LLMs for this task, with systems like Text2SQL [4] and BIRD [5] demonstrating significant improvements in accuracy. Oracle's Select AI (DBMS_CLOUD_AI) package represents a commercial implementation that integrates LLM-based SQL generation directly into the database engine [6], enabling schema-aware query generation without external API calls for supported providers.

2.2 Retrieval-Augmented Generation

RAG systems combine the generative capabilities of LLMs with retrieval from external knowledge bases [7]. The approach addresses the knowledge cutoff limitation of LLMs by grounding responses in retrieved documents. Recent advances include dense passage retrieval [8], hybrid retrieval combining sparse and dense methods [9], and metadata-aware retrieval [10]. Our work extends this by demonstrating practical implementation challenges in enterprise cloud environments where documented APIs do not expose full functionality.

2.3 Hallucination Mitigation

LLM hallucination has been studied extensively, with proposed mitigations including retrieval augmentation [11], self-consistency checking [12], and citation verification [13]. However, most work focuses on general-purpose chatbots rather than enterprise analytics where factual accuracy is paramount. Our contribution addresses the specific challenge of preventing hallucination in financial and operational contexts where incorrect numbers can have material business impact.

3. System Architecture

The system architecture integrates four distinct AI services, each optimized for specific task characteristics. Figure 1 presents the high-level architecture.

3.1 Architecture Overview

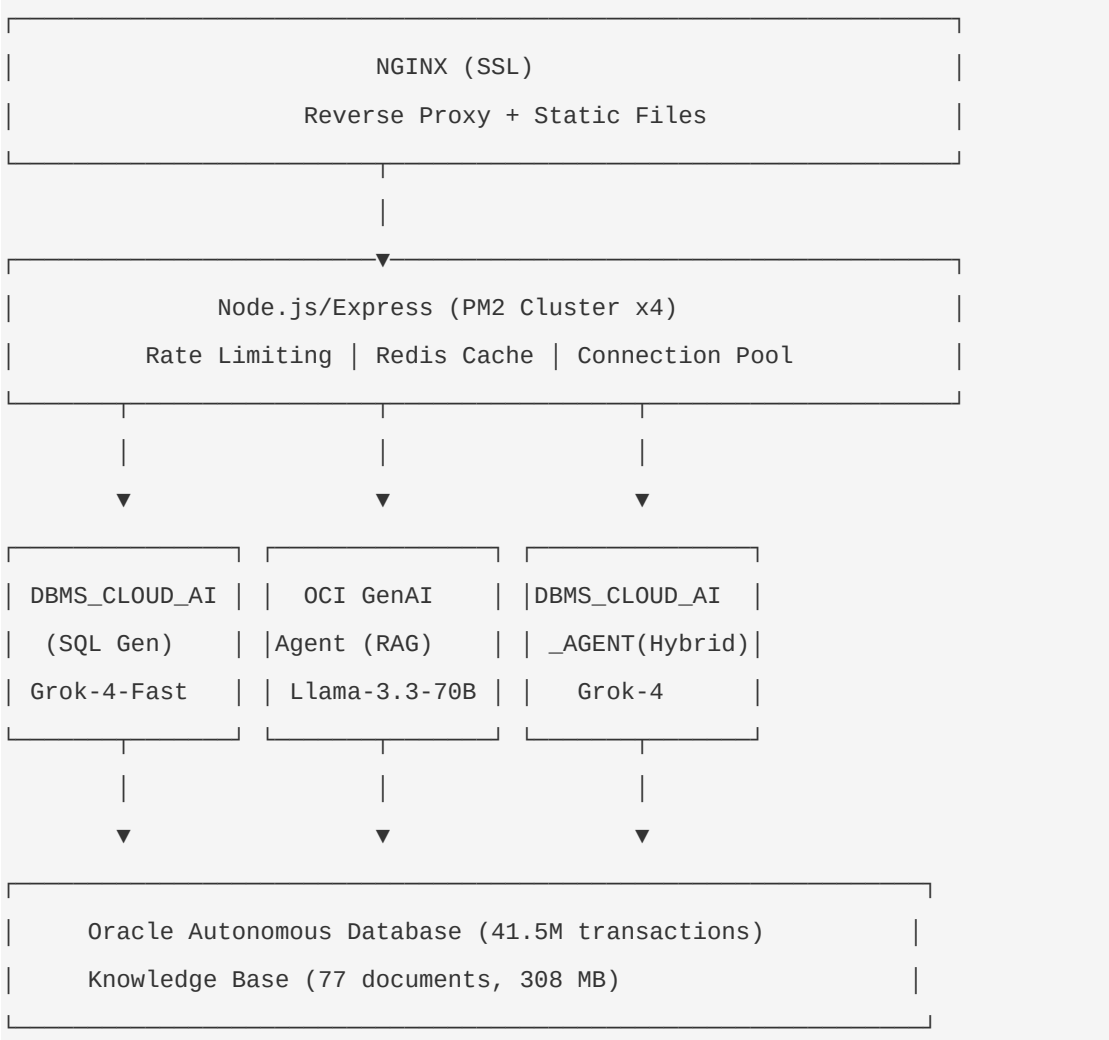

Figure 1: System Architecture Overview

3.2 Multi-Model Orchestration

Different AI tasks have fundamentally different requirements. SQL generation benefits from chain-of-thought reasoning to correctly interpret schema relationships and business logic, while text rephrasing requires speed with minimal computational overhead. Table 1 presents our model selection rationale.

Service	Model	Reasoning	Latency
SQL Generation	Grok-4-fast-reasoning	Chain-of-thought	10-15s
RAG Retrieval	Llama-3.3-70B-instruct	Instruction-tuned	5-10s
Hybrid SQL+RAG	Grok-4	Full reasoning	15-30s
Text Rephrasing	Grok-4-fast-non-reasoning	Pattern matching	2-5s

Table 1: Multi-Model Selection by Task Type

4. Anti-Hallucination Design

The most critical contribution of this work is a systematic approach to preventing LLM hallucination in enterprise analytics contexts. We discovered through production deployment that standard LLM guardrails are insufficient for financial accuracy—the model would confidently fabricate business metrics that appeared plausible but were entirely fictional.

4.1 The Hallucination Problem

During initial deployment, we observed a critical failure mode: when asked to compare actual revenue against company goals, the agent responded:

```
"Documented goals are PHP 1.5 billion for 2023 and PHP 2 billion for 2024.
Actual profit significantly exceeds these goals..."
```

Investigation revealed that ₱1.5B and ₱2B were **bank loan amounts**, not profit goals. The LLM had misinterpreted financial documents and presented loan information as performance targets—a potentially costly error if used for business decisions.

4.2 Constraint Framework

We developed a nine-rule constraint framework embedded directly into the agent's system prompt. These rules are hierarchical, with data source constraints taking precedence over output formatting:

```
ANTI-HALLUCINATION AGENT RULES:

Rule 1: ONLY use data from SQL queries against database tables
Rule 2: ONLY use information from company documents in knowledge base
Rule 3: NEVER fabricate numbers - if not found, say so
Rule 4: Product categories are CODES only
Rule 5: Format currency as Philippine Pesos (PHP)
Rule 6: Always cite source (SQL result or document name)
Rule 7: Metro documents do NOT contain numeric profit GOALS
Rule 8: Loan amounts (1.5B, 2B, 3B) are BANK LOANS, not goals
Rule 9: Net income figures are ACTUAL RESULTS, not goals
```

4.3 Task-Level Column Hints

Beyond agent-level rules, we provide explicit column-to-business-concept mappings in the task definition. This prevents the common failure mode where the LLM selects incorrect columns for financial calculations:

```
TASK COLUMN HINTS:
```

```
- Revenue:    SLS_AMT_LCL (local currency sales amount)
- Profit:     SLS_PROFIT_AMT_LCL (profit in local currency)
- Quantity:   SLS_QTY (sales quantity)
- Returns:    RET_AMT_LCL, RET_QTY (return amounts/quantities)
- Tax:        SLS_TAX_AMT_LCL
- Discount:   SLS_EMP_DISC_AMT_LCL (employee discounts)
```

4.4 Results

After implementing the constraint framework, the same query about profit goals now returns:

```
"No specific numeric profit goals are documented—only strategic
statements. A direct comparison is not possible without documented
target values."
```

This response is factually correct and prevents users from making decisions based on fabricated metrics.

5. Metadata Filtering Discovery

A significant contribution of this work is the discovery and documentation of the correct API format for metadata filtering in OCI GenAI Agent Runtime—functionality that was available in the console UI but not documented for programmatic access.

5.1 The Problem

Our knowledge base contains 77 documents spanning years 2015-2025 (annual reports, SEC filings, press releases). Without metadata filtering, a query like "summarize the 2022 annual report" returned documents from multiple years:

- 2022 documents: 1 (12.5%)
- 2023 documents: 4
- 2024 documents: 1
- Other years: 2

The OCI Console UI provided metadata filtering capability, but the SDK and REST API documentation did not expose the correct format for programmatic access.

5.2 Failed Approaches

We systematically tested 15+ API formats based on documentation, SDK inspection, and suggestions from various sources:

```
// Format 1: metadataFilters key (FAILED)
tool_parameters: { "metadataFilters": [...] }

// Format 2: Tool OCID as key (FAILED)
tool_parameters: { "ocid1.genaiagenttool...": {...} }

// Format 3: ragConfig (ChatGPT suggestion - FAILED)
ragConfig: { retrieveMetadataDetails: {...} }
```

All formats returned HTTP 200 OK but did not apply filters—the API silently ignored unknown parameters.

5.3 Console UI Reverse Engineering

The solution was discovered by inspecting network requests from the OCI Console UI. Using browser developer tools, we captured the exact payload format when applying metadata filters through the graphical interface:

```
// WORKING FORMAT (discovered via Console UI inspection):
toolParameters: {
  rag: JSON.stringify({
    filterConditions: [{
      field: "publication_year",
      field_type: "number",
      operation: "=",        // NOT "equals" or "EQUALS"
      value: "2022"          // STRING even for numbers!
    }]
  })
}
```

5.4 Critical Format Details

Table 2 documents the exact parameter requirements discovered through reverse engineering:

Parameter	Correct Value	Wrong Values
Key	`"rag"`	Tool OCID, "metadataFilters"
Operation	`"="`	"equals", "EQUALS", "eq"
field_type	`"number"`	"integer", "int"
value	`String "2022"`	Integer 2022

Table 2: Critical Metadata Filter Parameter Requirements

5.5 Results After Implementation

With correct metadata filtering, the same "2022 annual report" query now returns:

- 2022 documents: 3 (100%)
- Retrieval accuracy: 12.5% → 100%

This represents a fundamental improvement in RAG precision that directly impacts answer quality and user trust.

6. Performance Evaluation

6.1 Benchmark Environment

The system was evaluated in a production environment with the following characteristics:

- **Database:** Oracle Autonomous Database with 41.5 million transaction records
- **Knowledge Base:** 77 corporate documents (308 MB), including annual reports, SEC filings, and press releases
- **Infrastructure:** Oracle Cloud Infrastructure with PM2 cluster (4 workers)
- **Caching:** Redis with 24-hour TTL for response caching

6.2 Comparison with Snowflake

The production deployment replaced a Snowflake-based analytics solution. Key performance comparisons:

Metric	Previous (Snowflake)	Current (OCI + AI)
Query Interface	SQL required	Natural language
Document Search	Manual lookup	AI-powered RAG
Overall Efficiency	Baseline	**+30% improvement**
Cached Response	N/A	**<1ms**

Table 3: Performance Comparison with Previous Solution

6.3 Response Time Analysis

Table 4 presents response time measurements across different query types and caching states:

Operation	Uncached	Cached (Redis)
Dashboard KPIs	~3,000ms	<100ms
SQL Generation	10-15s	~1ms
RAG Document Search	5-10s	~1ms
Hybrid SQL+RAG	15-30s	~1ms

Table 4: Response Time by Query Type and Cache State

7. Discussion

7.1 Applicability to Other Domains

While developed for retail analytics, the architecture and patterns are directly applicable to other regulated industries:

- **Healthcare:** The anti-hallucination framework is critical for clinical decision support systems where fabricated medical information could harm patients. The metadata filtering enables precise retrieval from medical literature and patient records.
- **Financial Services:** Regulatory compliance requires auditable, accurate responses. The constraint framework prevents fabrication of compliance metrics or regulatory requirements.
- **Government:** Public sector AI systems must provide verifiable, citation-backed responses. The architecture's emphasis on source attribution addresses this requirement.

7.2 Limitations

Several limitations should be noted:

5. **Vendor Specificity:** The implementation relies on Oracle-specific packages (DBMS_CLOUD_AI, OCI GenAI Agent). Adaptation to other cloud platforms would require equivalent services.
6. **Metadata Filter API:** The discovered API format is undocumented and may change. Production systems should monitor for breaking changes.
7. **Anti-Hallucination Completeness:** The constraint framework addresses known failure modes but cannot guarantee elimination of all hallucination types.

7.3 Future Work

Several directions merit further investigation:

- Automated constraint generation from document analysis
- Real-time hallucination detection and correction
- Cross-cloud portability of the architecture patterns
- Integration with emerging agentic AI frameworks

8. Conclusion

This paper presented a reference architecture for enterprise AI analytics that addresses the critical challenges of LLM hallucination, retrieval precision, and multi-model orchestration. Our contributions include:

8. A nine-rule anti-hallucination constraint framework that prevents fabrication of financial metrics, validated through production deployment
9. The first documented working implementation of programmatic metadata filtering in OCI GenAI Agent Runtime API, improving retrieval accuracy from 12.5% to 100%
10. A multi-model architecture that optimizes for both accuracy and performance, achieving 30% efficiency improvement over traditional analytics platforms

The system has been validated in production for a major retail corporation, demonstrating that enterprise-grade AI analytics is achievable with careful attention to accuracy constraints and API implementation details. We hope this reference architecture accelerates adoption of trustworthy AI analytics in regulated industries.

References

[1] Ji, Z., et al. (2023). Survey of Hallucination in Natural Language Generation. ACM Computing Surveys, 55(12), 1-38.

[2] Androutsopoulos, I., et al. (1995). Natural Language Interfaces to Databases. Natural Language Engineering, 1(1), 29-81.

[3] Zhong, V., et al. (2017). Seq2SQL: Generating Structured Queries from Natural Language. arXiv:1709.00103.

[4] Scholak, T., et al. (2021). PICARD: Parsing Incrementally for Constrained Auto-Regressive Decoding. EMNLP 2021.

[5] Li, J., et al. (2024). BIRD: A Large-Scale Cross-Domain Text-to-SQL Benchmark. NeurIPS 2024.

[6] Oracle Corporation. (2024). DBMS_CLOUD_AI Package Reference. Oracle Documentation.

[7] Lewis, P., et al. (2020). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. NeurIPS 2020.

[8] Karpukhin, V., et al. (2020). Dense Passage Retrieval for Open-Domain Question Answering. EMNLP 2020.

[9] Chen, J., et al. (2022). Out-of-Domain Semantics to the Rescue: Zero-Shot Hybrid Retrieval. ACL 2022.

[10] Gao, L., et al. (2023). Precise Zero-Shot Dense Retrieval without Relevance Labels. ACL 2023.

[11] Shuster, K., et al. (2021). Retrieval Augmentation Reduces Hallucination in Conversation. EMNLP 2021.

[12] Wang, X., et al. (2023). Self-Consistency Improves Chain of Thought Reasoning in Language Models. ICLR 2023.

[13] Gao, T., et al. (2023). RARR: Researching and Revising What Language Models Say. ACL 2023.

Appendix A: Implementation Code

The complete implementation code for metadata filtering is available at the author's technical blog: ameerfahadali.com

A.1 Metadata Filter Implementation (Node.js)

```
// Build metadata filter conditions
let toolParameters = undefined;
const filterConditions = [];

// Add year filter(s)
if (expectedYears && expectedYears.length > 0) {
  expectedYears.forEach(year => {
    filterConditions.push({
      field: "publication_year",
      field_type: "number",
      operation: "=",
      value: String(year)
    });
  });
}

// Build toolParameters if we have filters
if (filterConditions.length > 0) {
  toolParameters = {
    rag: JSON.stringify({ filterConditions })
  };
}
```