When RAG Hallucinates Numbers

Graph-RAG for Precise Answers

Based on: MetaRAG (arxiv 2509.09360) and RAG-KG-IL (arxiv 2503.13514)

Elizabeth Fuentes Leone

Developer Advocate @ AWS

elifuentes.tech

6 Techniques to Stop AI Agents from Failing

This talk is part of a series covering production patterns for reliable AI agents:

01 **Graph-RAG**

Knowledge graphs for grounded answers

02 **Semantic Tool Selection**

FAISS filtering to reduce tokens and errors

03 **Multi-Agent Validation**

Executor, Validator, Critic pipeline

04 **Neurosymbolic Guardrails**

Hard rules the LLM cannot bypass

05 **Agent Control Steering**

Self-correction instead of failure

06 **Production Deployment**

MCP Gateway, serverless, observability

Today: Graph-RAG for Precise Answers

01 **Graph-RAG**

Knowledge graphs for grounded answers

02 **Semantic Tool Selection**

FAISS filtering to reduce tokens and errors

03 **Multi-Agent Validation**

Executor, Validator, Critic pipeline

04 **Neurosymbolic Guardrails**

Hard rules the LLM cannot bypass

05 **Agent Control Steering**

Self-correction instead of failure

06 **Production Deployment**

MCP Gateway, serverless, observability

Your RAG Agent Seems Smart...

...until you ask it to count something.

"How many hotels have a swimming pool?"

RAG says: "approximately 45-50"

Reality: 133

Fabricated Statistics

Invents counts and averages from 3 retrieved chunks

Incomplete Retrieval

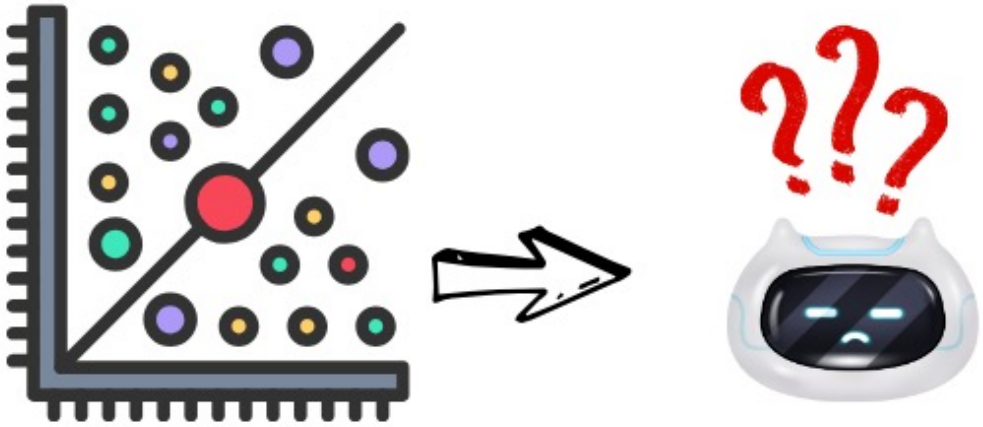
Sees 3 of 300 documents, generalizes from fragments

? Out-of-Domain Fabrication

Vector search always returns results, even for Antarctica

RAG vs Graph-RAG: The Fundamental Difference

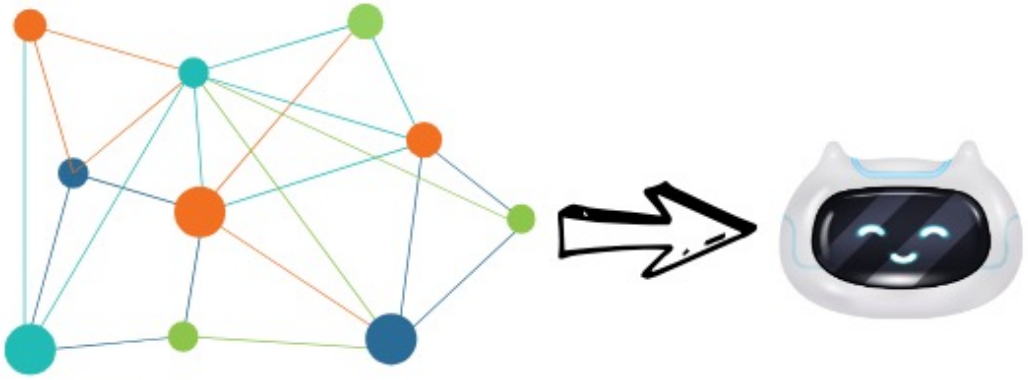
Agentic RAG



Test → Performance

Multi-step reasoning → ❌ Cannot aggregate
Comparative queries → ⚠️ May hallucinate stats
Exact counts → ❌ Partial results only
Out-of-domain → ⚠️ Returns similar results

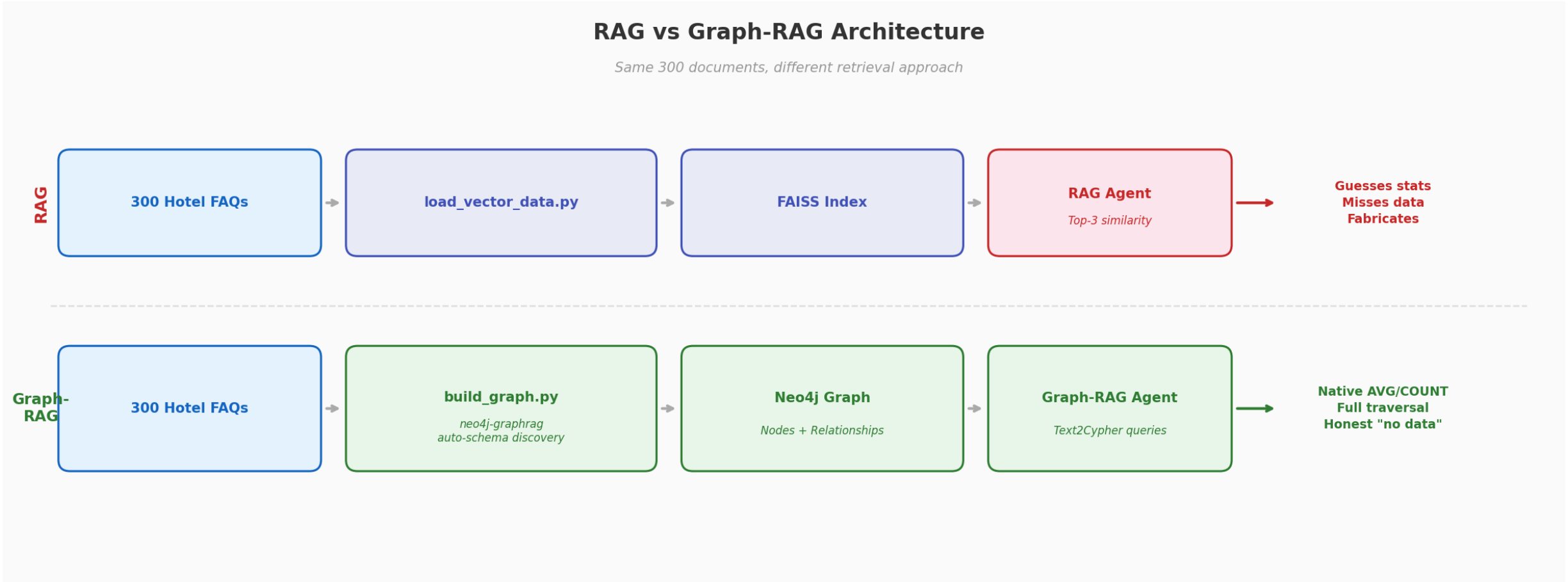
Agentic Graph-RAG



Test Performance

Multi-step reasoning → ✅ Uses relationship traversal
Comparative queries → ✅ Precise aggregation
Exact counts → ✅ Complete filtering
Out-of-domain → ✅ Explicit failure

Same Data, Different Architecture



Both agents receive the same 300 hotel FAQ documents and the same queries.

Building the Knowledge Graph

`build_graph.py — neo4j-graphrag SimpleKGPipeline`

1

No Manual Schema

LLM reads each document and discovers entity types automatically.
Hotel, Room, Amenity, Policy, Service
emerge from the documents.

2

Entity Resolution

Deduplication merges synonyms:
"Outdoor Pool" + "Swimming Pool"
→ one Amenity node.
This is what makes COUNT() exact.

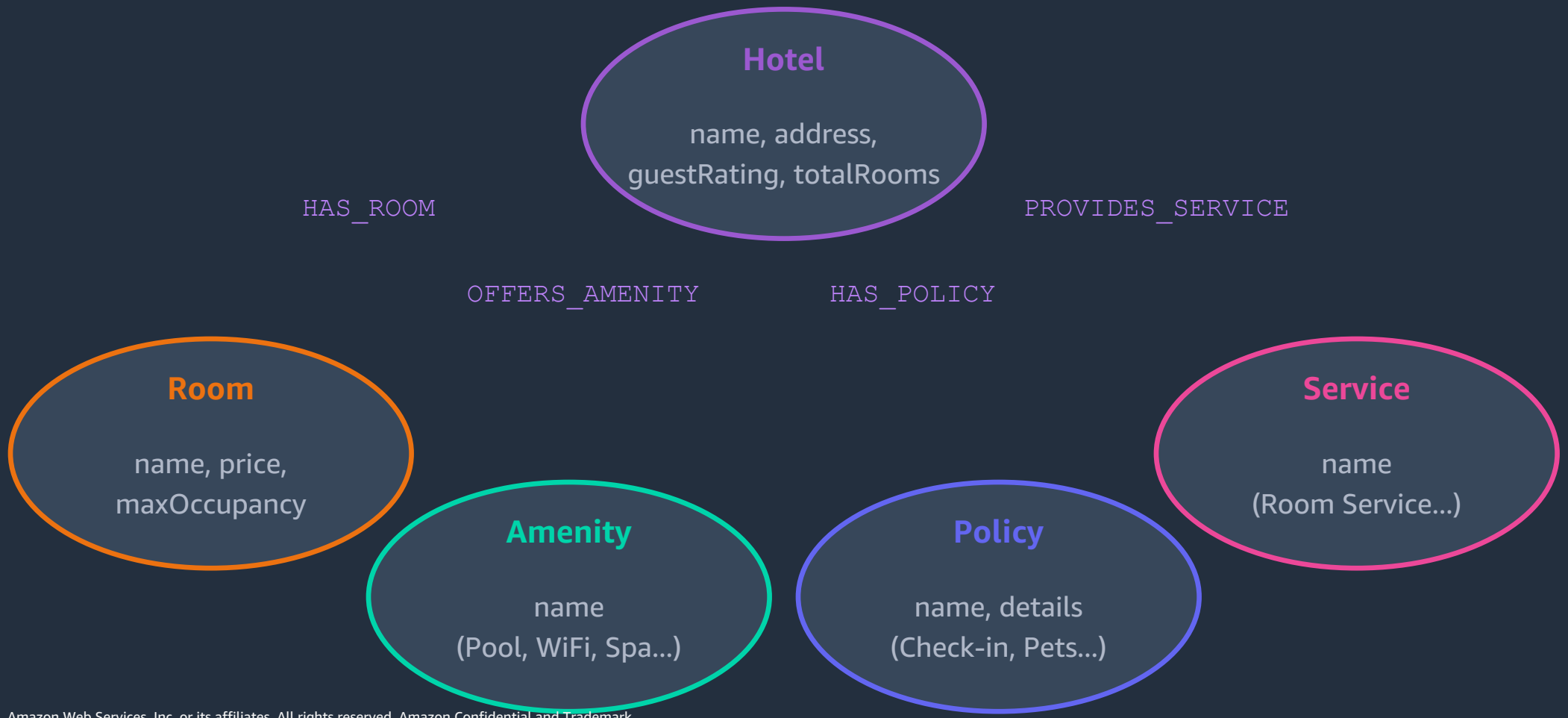
3

One-Time Build Cost

Lite (30 docs): ~15 minutes.
Full (300 docs): ~2 hours.
After that: every query is instant and deterministic.

Auto-Generated Knowledge Graph

No manual schema design. neo4j-graphrag discovers entities automatically.



The @tool Pattern: Schema as Grounding

The schema lives in the tool docstring — not in the prompt.

```
@tool
def query_graph(question: str) -> str:
    """Query the hotel knowledge graph.
    Schema:
      Nodes: Hotel, Room,
             Amenity, Policy, Service
      Rels:  HAS_ROOM, OFFERS_AMENITY,
             HAS_POLICY, PROVIDES_SERVICE
      Hotel: name, address,
             guestRating, totalRooms
    """
    return neo4j.run(
        text2cypher(question))
```

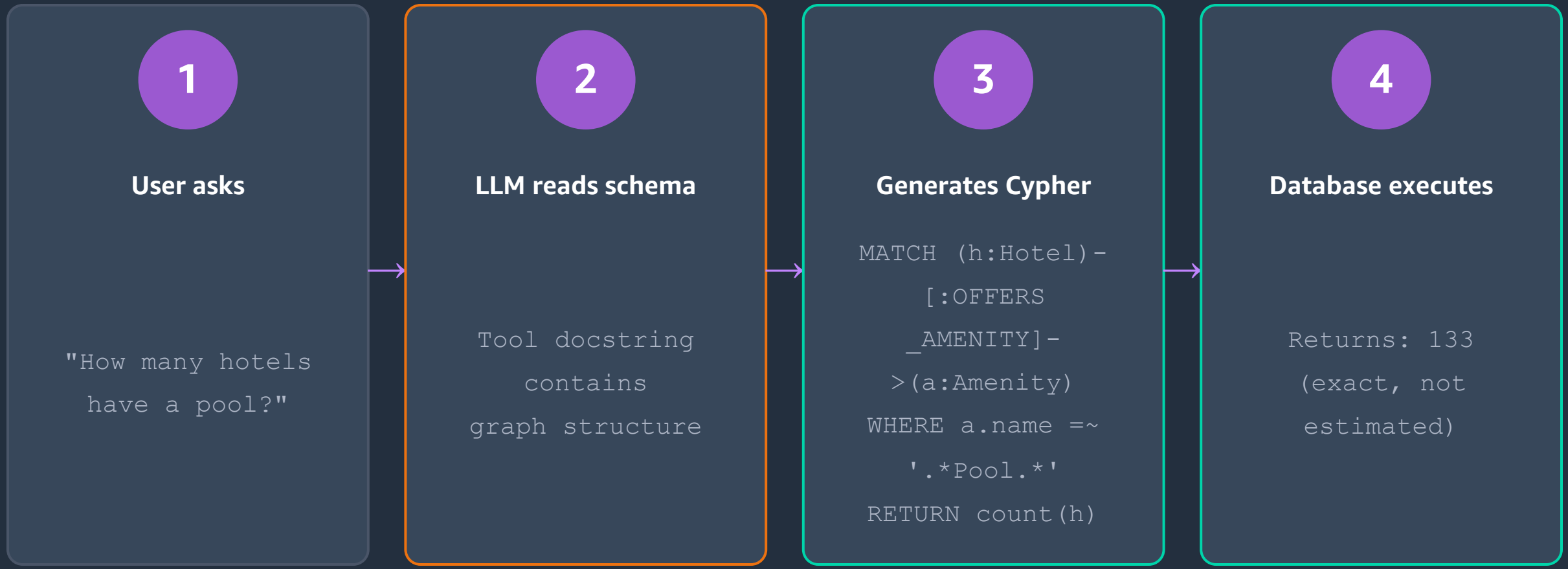
GROUNDED **LLM cannot invent node types**

The schema in the docstring tells it exactly which nodes and relationships exist. It cannot query a type that is not listed.

HONEST **Empty = honest answer**

The tool returns "No results found" when the database has nothing. The LLM reports that verbatim. No filling the gap.

Text2Cypher: Natural Language to Precise Queries



The LLM cannot fabricate: the database either has the data or it doesn't.

Demo Time

4 queries. Same data. RAG vs Graph-RAG.

Test 1: Aggregation

"What is the average guest rating across all hotels in Paris?"

RAG

4.7

Manually calculated from
2 retrieved documents

Graph-RAG

4.7

Native AVG() in Cypher
Database-level computation

Same answer, different reliability. RAG got lucky with 2 docs.

Test 2: Precise Counting

"How many hotels have a swimming pool as an amenity?"

RAG

"I don't have the data"

Only sees 3 of 300 documents.
Cannot count across full dataset.

Graph-RAG

133

Exact COUNT via Cypher.
Complete filtering across all nodes.

Vector similarity fundamentally cannot count.

Test 3: Multi-Hop Reasoning

"What are the room types and prices for the highest rated hotel?"

RAG

Found 1 hotel but no rooms

Cannot traverse relationships.
Rating and rooms are in
different text chunks.

Graph-RAG

Complete room data

Traversed Hotel -> Room nodes.
ORDER BY guestRating DESC
LIMIT 1 + relationship traversal.

Knowledge graphs preserve relationships that chunking destroys.

Test 4: Out-of-Domain Detection

"Tell me about hotels in Antarctica"

RAG

HALLUCINATED

- Research Stations
- Expedition Cruises
- Specialized Lodges

None of these exist in the dataset.

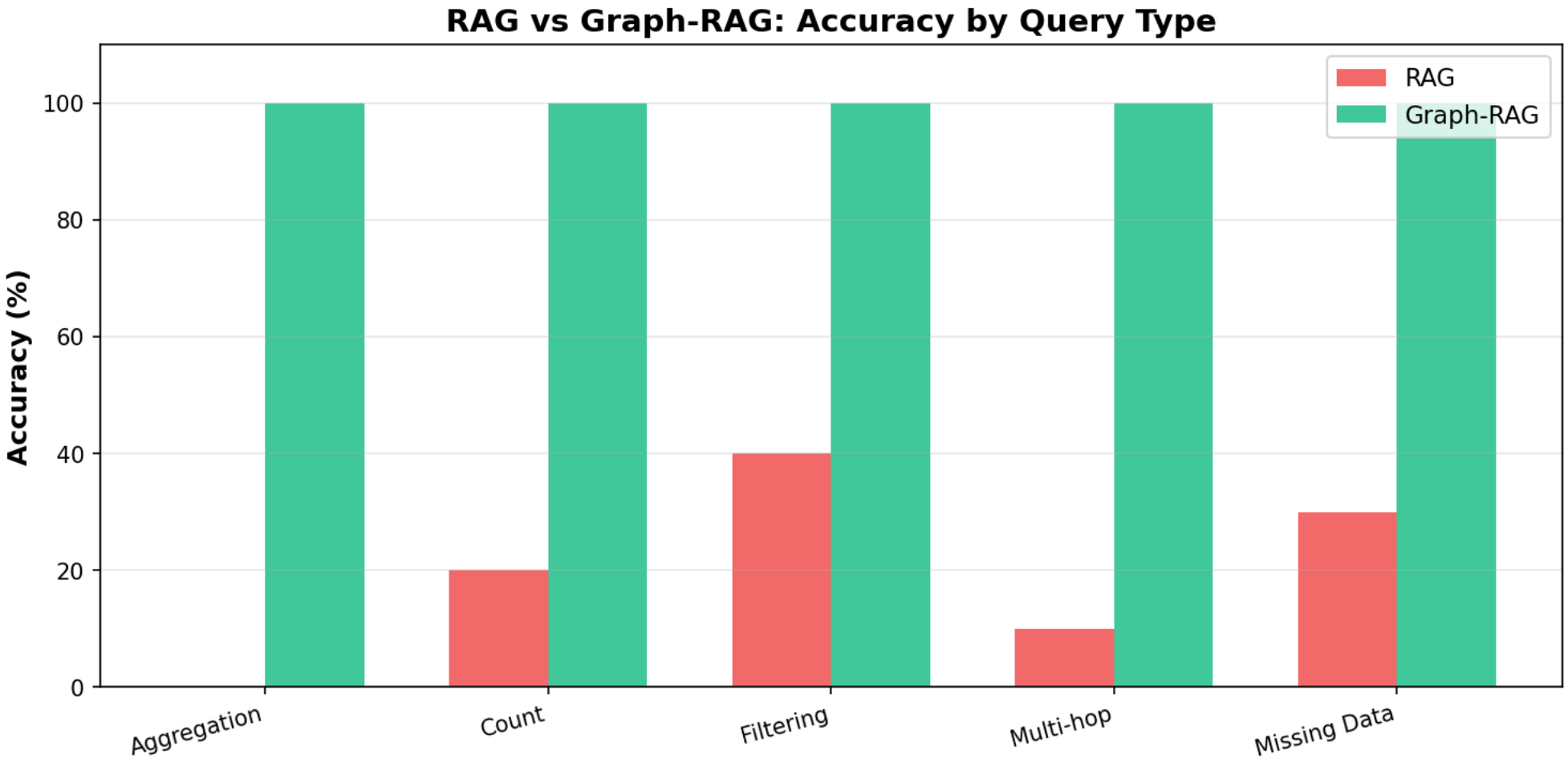
Graph-RAG

"No hotels listed in Antarctica"

Returns empty result set.
Honest failure, not fabrication.

RAG always returns something. Graph-RAG returns empty when data doesn't exist.

RAG vs Graph-RAG: Accuracy by Query Type



Graph-RAG: 100% across all query types. RAG: 0-40%.

Production Patterns

1 @tool Decorator

Wrap Cypher queries as agent tools.
Schema in docstring grounds the LLM.

2 Hybrid Architecture

Vector search for open-ended questions.
Graph for structured queries (counts, filters).

3 Performance

Graph build: one-time cost.
Query accuracy: deterministic, auditable.

4 Error Handling

Empty results = honest "no data".
No silent fabrication.

When to Use RAG vs Graph-RAG

Use Graph-RAG

- ✓ Precise queries (exact counts, filters)
- ✓ Aggregations (AVG, SUM, COUNT)
- ✓ Multi-hop reasoning (relationships)
- ✓ Structured data with clear schemas
- ✓ Verifiable, auditable results

Use RAG

- Semantic search (similar concepts)
- Unstructured text (articles, docs)
- Fuzzy matching (approximate)
- Simple retrieval (Q&A lookups)
- No graph infrastructure needed

Key Takeaways

1. Implement Graph-RAG with Neo4j and auto entity extraction for any document set
2. Apply Text2Cypher query generation for precise, unfabricable answers
3. Build a concrete decision framework for when to use RAG vs Graph-RAG
4. Design hybrid architectures: Graph-RAG for structured, RAG for unstructured
5. Evaluate open-source tools adaptable to any domain with structured data

Resources

Code	github.com/elizabethfuentes12/why-agents-fail-sample-for-amazon-agentcore
Paper	MetaRAG: Meta-analysis of RAG failure modes (arxiv 2509.09360)
Paper	RAG-KG-IL: RAG with Knowledge Graph Integration (arxiv 2503.13514)
Blog	dev.to/aws/rag-vs-graphrag-when-agents-hallucinate-answers

This is Demo 01 of 6

Up Next: Demo 02 — Semantic Tool Selection

PROBLEM

29 tools in your agent.

The agent picks the wrong one
or burns tokens reviewing all of them
before every call.

FIX

FAISS tool registry.

Filters 29 tools down to 5 relevant
ones before the agent sees them.
Fewer tokens. Fewer wrong calls.

Full code open-source — link in description.

Thank You!

Elizabeth Fuentes Leone

Blog & Socials

elifuentes.tech

Resources

bit.ly/4uYWb0R