The Three Hidden Graphs in Every AI Agent

What it knows, what it remembers, and why it decided.

Elizabeth Fuentes Leone

Developer Advocate, GenAI · AWS

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.



Elizabeth Fuentes Leone

Developer Advocate @ AWS



elifuentes.tech



© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Three graphs hiding in every agent

Knowledge Graph

What it KNOWS

answers from connected facts, not similar text

"how many, on average?"

Graph Memory

What it REMEMBERS

follows the relationships between memories

"who connects to this?"

Reasoning Memory

WHY it decided

traces a decision back to its source

"why did you pick that?"

Three questions of the same agent: what it knows, what it remembers, and why it decided.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Knowledge Graph

Ask for a number, RAG makes one up

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

Vector search returns the 3 closest FAQ chunks, not all the hotels.
There is no AVG step, so the model averages those 3. The number is fabricated.

Vector RAG retrieves by similarity. It ranks text; it never computes an aggregation.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Knowledge Graph

The graph computes the answer, or says none

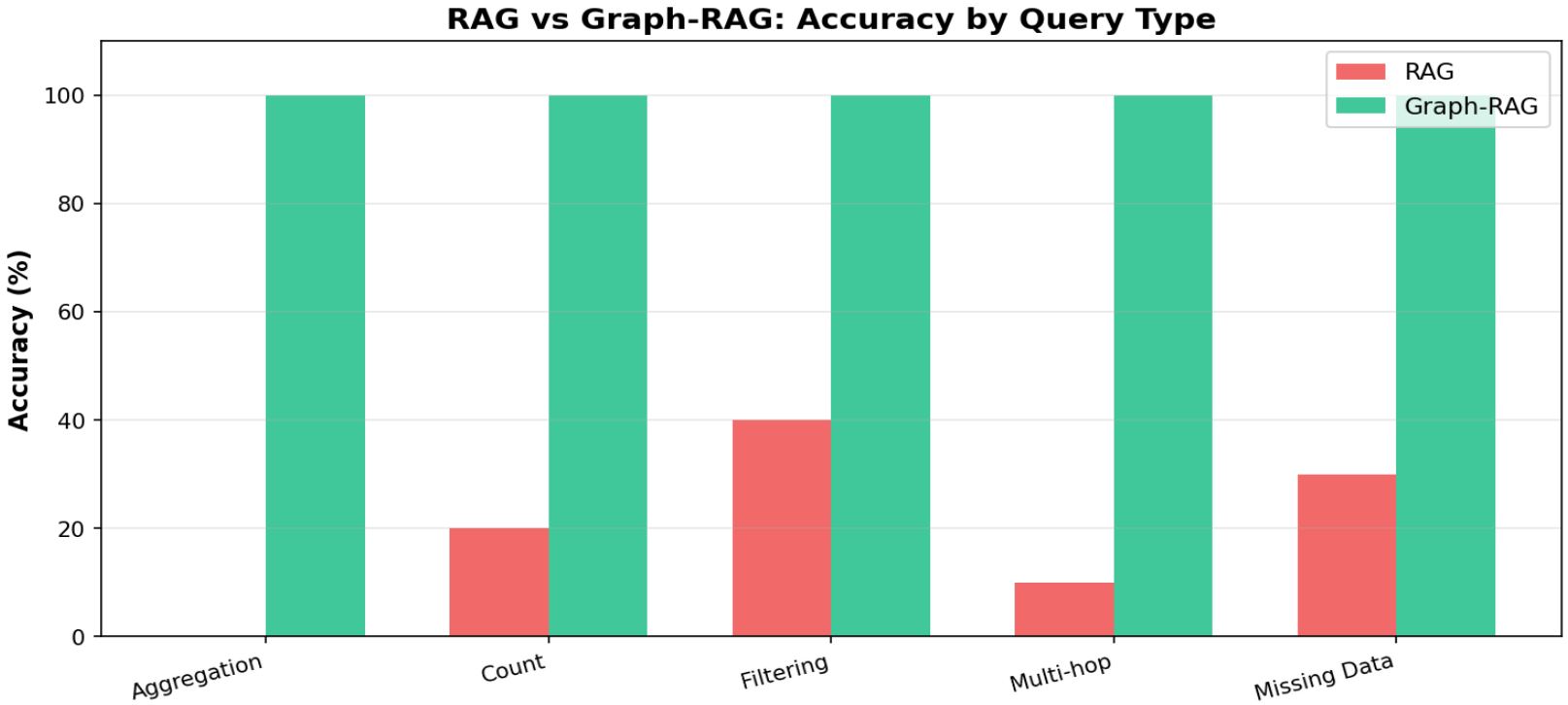
Question	Vector RAG	Graph-RAG
average rating	top-3 chunks, no math	**AVG() on the graph**
how many have a pool	sees only top-k docs	**exact COUNT() across all**
hotels in Antarctica	fabricates something	**"no results", honest**

Neo4j: a knowledge graph retrieves grounded, traceable context instead of guessing.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Knowledge Graph

Measured on 300 hotel FAQs

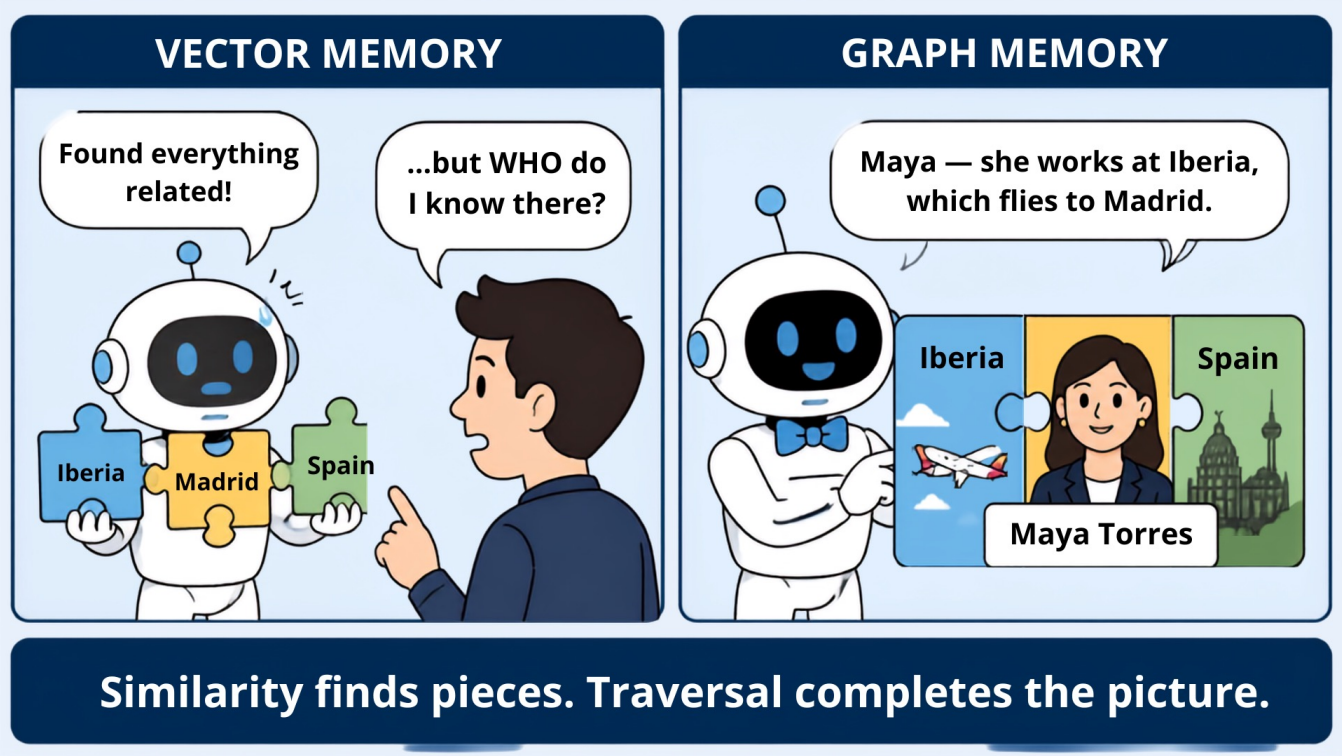


The knowledge base as a graph: aggregations, counts, and honest 'no results'. Run VISUALIZE_QUERIES to see it.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Graph Memory

It knows the pieces, not the person



"Who do I know connected to flights to Spain?" Vector memory returns Iberia, Madrid, Spain. Never the person.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Graph Memory

Vector, graph, or both

Vector search

similarity over embeddings; returns text, no traversal

Good for

- notes like this one
- what did I say about X

Graph traversal

you write the pattern; Cypher walks typed relations

Good for

- averages and counts
- walk known relations

Vector + graph

similarity finds the entry, then Cypher walks the relations

Good for

- multi-hop in plain language
- who connects to X

Both graph modes walk relations with Cypher. What differs is the entry point: an exact pattern, or similarity. GraphRAG puts the vector index inside the graph.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reasoning Memory

Ask it why a week later



The decision was real, made with actual tools and data. But the steps were never saved, so asking 'why?' later gets a reconstruction, not the record.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reasoning Memory

Why saving the trace matters

Audit	Reproduce	Reuse
prove why a decision was made, and trace what a bad source touched	the real chain on demand, not a story told after the fact	cache a past plan and skip the re-run: 86% less cost

A saved trace turns 'trust me' into something you can check, reuse, and audit.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reasoning Memory

The harness already emits it, a hook records it

```
class Neo4jDecisionRecorder(HookProvider):
    def register_hooks(self, registry):
        registry.add_callback(BeforeInvocationEvent, self._on_start)
        registry.add_callback(AfterToolCallEvent,   self._on_tool)
        registry.add_callback(AfterInvocationEvent, self._on_end)

    def _on_tool(self, event):
        name = event.tool_use["name"]         # the tool
        self._steps.append(name, event.tool_use["input"])
```

STRANDS AGENTS SDK

Real code from Demo 06. The harness already fires these events, so you add a listener, not new plumbing.

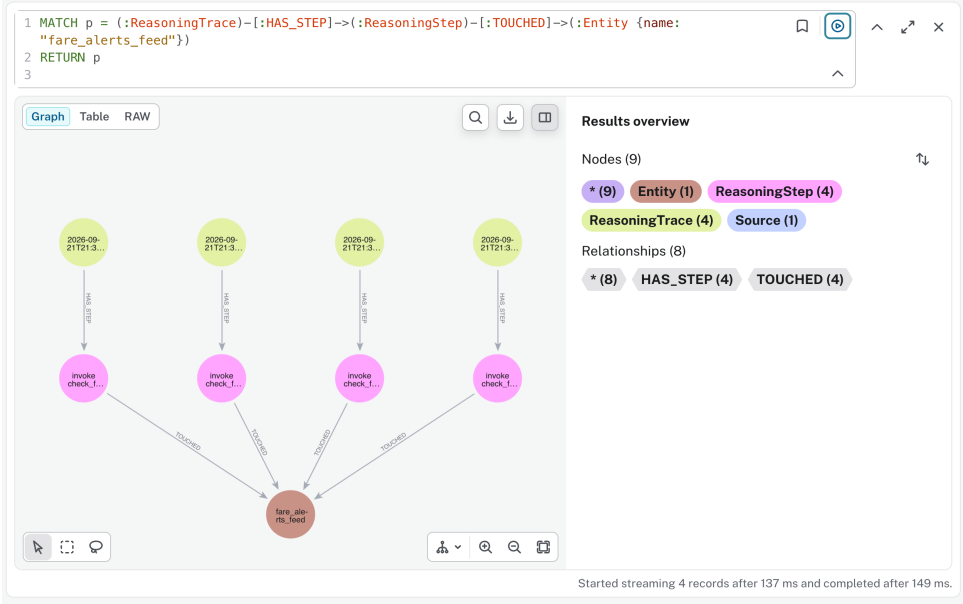
aws

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Reasoning Memory

A source went wrong. What did it touch?

"The fare-alerts feed was wrong. Which decisions touched it?"



One graph traversal, at read time. A flat store scans every record one at a time.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Why Strands: a few lines is the whole harness

```
agent = Agent(
  model=MODEL,
  tools=[
    recall_graph,
    remember_fact,
  ],
  hooks=[recorder],
)
```



- Building the agent is a few lines: a model, a list of tools, a hook.
- The hook listens to events the agent already emits: the tool name, input and result.
- The graphs fall out of what the agent already does. No tool rewrites.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Three questions your agent can finally answer

What it knows	a knowledge graph answers from connected facts; vector search returns similar text and lets the model fill gaps
What it remembers	graph memory follows the relationships between memories; vector search finds each on its own
Why it decided	record the steps, so 'why?' is a real answer and a bad source is one traversal away

Three graphs, three failures a plain store leaves unsolved.

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.

Thank you!

Elizabeth Fuentes Leone

Developer Advocate, GenAI · AWS

What it knows, what it remembers, why it decided.

Talk resources

Blog & Socials

elifuentes.tech

© 2026, Amazon Web Services, Inc. or its affiliates. All rights reserved.