ARTICLE

Graph RAG: when structure beats similarity

Graph retrieval solves one specific failure of vector search — questions whose answer is distributed across a corpus rather than sitting in any single passage. The cost that kept it out of reach has collapsed. What remains is a question about your users, not your budget.

PREPARED BY

Junior Williams
Founder, Junior Williams Consulting Inc.

DATE

26 July 2026

CONTACT

Book an intro call
trustcyber.ca

Bottom line up front. Graph retrieval-augmented generation (Graph RAG) answers one class of question that ordinary vector retrieval cannot: questions whose answer is spread across a corpus rather than located in a passage. It costs more to build, it is fragile in exactly one place — entity identity — and until eighteen months ago the price put it out of reach for most teams. That constraint has collapsed. The decision now turns on the shape of your questions.

1. The question vector search cannot answer

Retrieval-augmented generation in its standard form is a nearest-neighbour lookup. You embed the corpus into chunks, embed the query, return the top *k* by cosine similarity, and hand the winners to the model. It works well and it is cheap. Its limitation is structural rather than a matter of tuning: the retrieved chunks arrive as unrelated islands. Nothing in the index records that the incident report in chunk 4,102 concerns the same component as the procurement email in chunk 88,417.

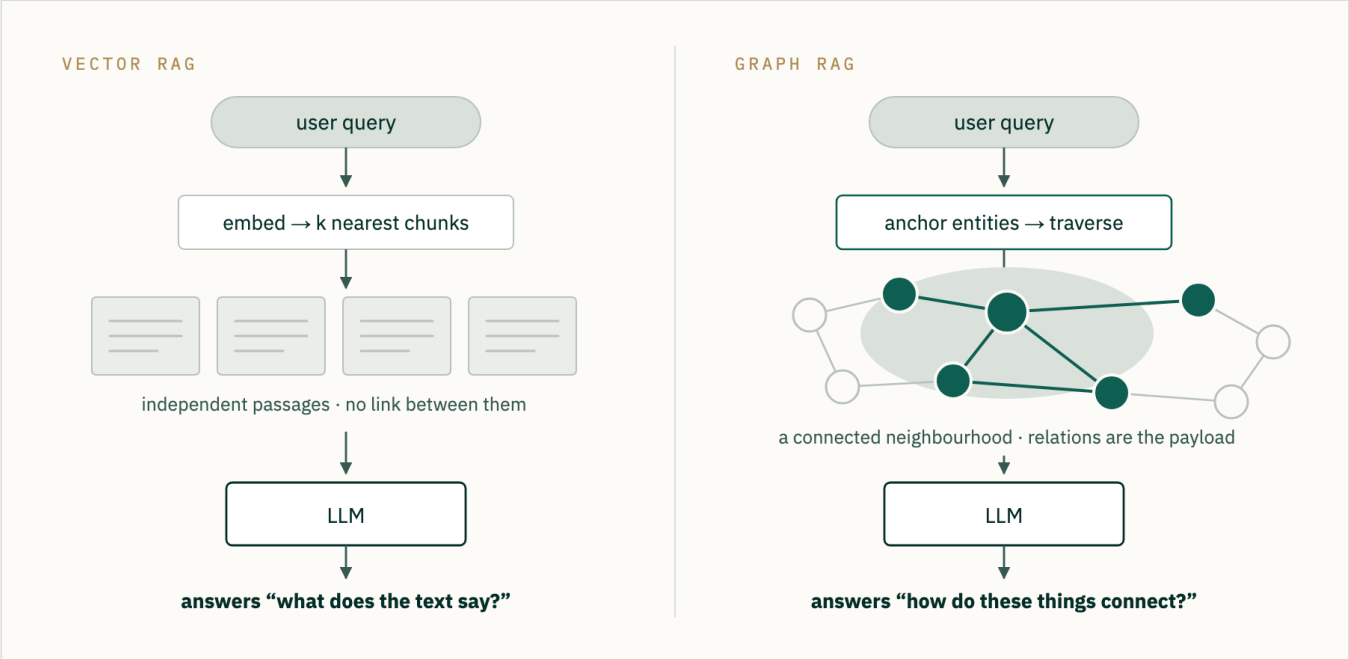

FIGURE 1 Vector retrieval returns the nearest passages. Graph retrieval returns a connected neighbourhood, and the connections are part of what reaches the model.

That distinction matters for two families of query. The first is multi-hop: *which supplier is exposed to the outage in the Rotterdam facility?* The answer requires chaining facility to component to part number to supplier, and no single chunk holds the chain. The second is corpus-wide, sometimes called query-focused summarisation: *what are the recurring themes across five years of maintenance reports?* Here similarity search fails on a different axis. There is no passage to be nearest to, because the answer is a property of the whole collection.

The GraphRAG paper from Microsoft Research frames the second case precisely. Its target is queries requiring "understanding of semantic concepts over entire text corpora" — a class it argues vector retrieval cannot serve at all.

2. How it works

Every Graph RAG system splits into three stages, a decomposition the field's main survey formalises as graph-based indexing, graph-guided retrieval, and graph-enhanced generation. The interesting variation lives in the first stage.

Index time

Microsoft's reference pipeline runs raw text through a language model twice. The first pass extracts entities, relationships, and claims from each chunk; those get deduplicated and merged into a knowledge graph. The second pass detects communities in that graph and writes natural-language summaries of each one at several levels of granularity.

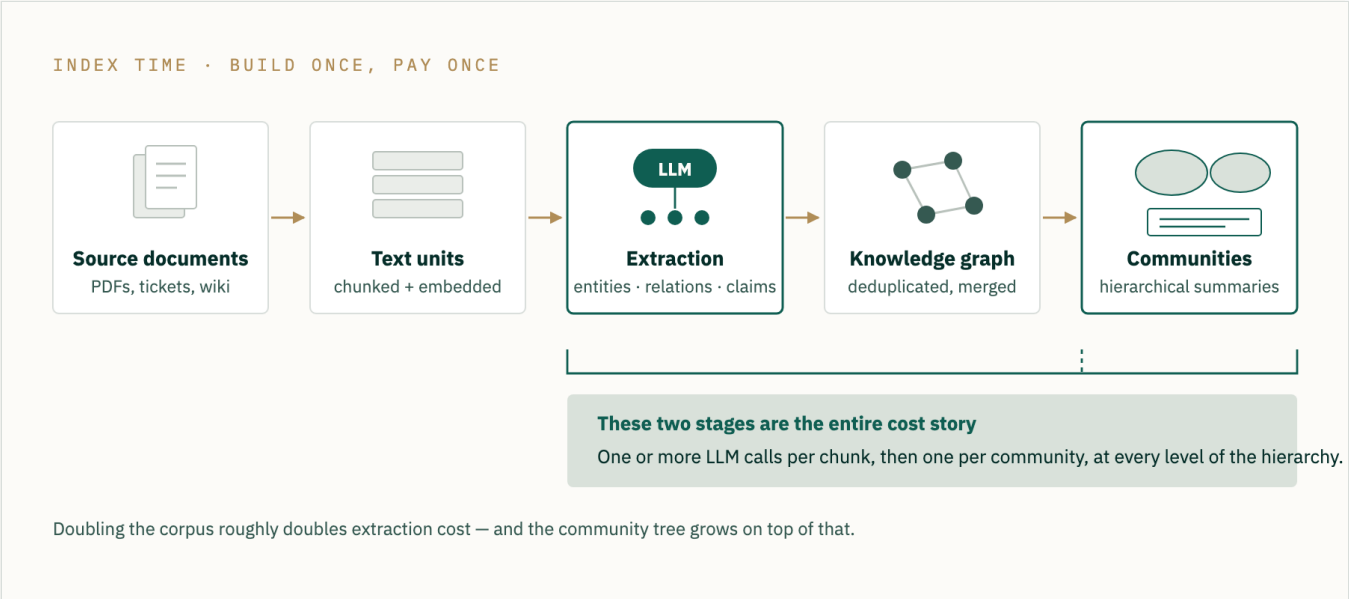

FIGURE 2 The indexing pipeline. Only two stages call a language model, and those two stages are the whole cost story.

Community detection uses the Leiden algorithm, applied recursively — the paper describes "recursively detecting sub-communities within each detected community until reaching leaf communities that can no longer be partitioned." The choice is deliberate. Graphs built from real text are modular, and that modularity yields a free thematic partition of the corpus at multiple zoom levels.

Query time

Two retrieval modes fall out of the structure. **Local search** anchors on entities named in the query, walks outward across the graph, and assembles the neighbourhood — entities, their relationships, and the source

text units each was extracted from — into the prompt. This is the mode for multi-hop questions. **Global search** ignores the raw text entirely. It fans the query across the pre-written community summaries, generates a partial answer from each, scores them, and reduces the survivors into a final response.

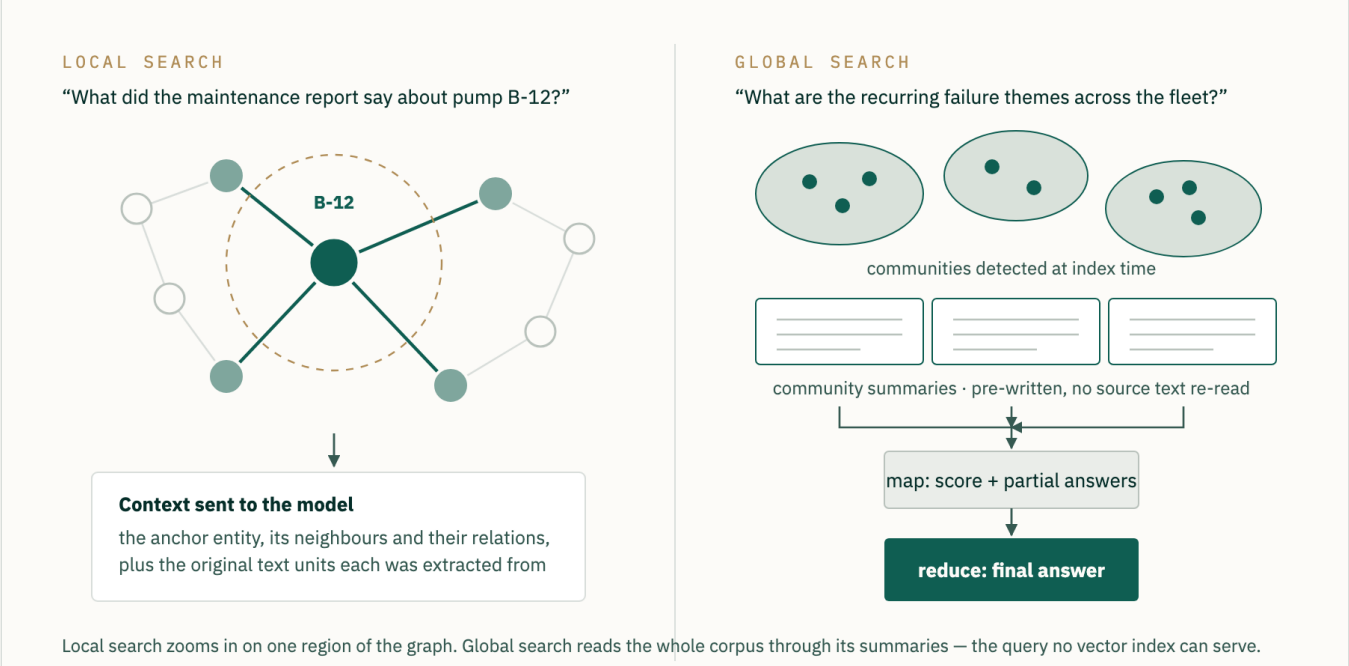

FIGURE 3 Local search zooms in on one region of the graph. Global search reads the entire corpus through its summaries.

The map-reduce structure is what makes corpus-wide questions tractable, and the efficiency gain is substantial. On the paper's news dataset, answering from root-level community summaries consumed 39,770 context tokens against 1,707,694 for summarising the source text directly — roughly a 43-fold reduction, while scoring better on comprehensiveness and diversity.

3. The cost problem, and its collapse

The catch was always index construction. Extraction runs at least one model call per chunk, and community summarisation adds another per community per hierarchy level. Cost scales with corpus size and then some, which is why early GraphRAG deployments were priced like infrastructure projects rather than features.

LazyGraphRAG removed that constraint by inverting the design. Instead of calling a language model at index time, it uses noun-phrase extraction to build a concept co-occurrence graph, applies graph statistics to extract the community hierarchy, and defers every model call to query time. Retrieval then combines best-first and breadth-first search in an iterative-deepening loop, using a sentence-level relevance assessor with a configurable budget.

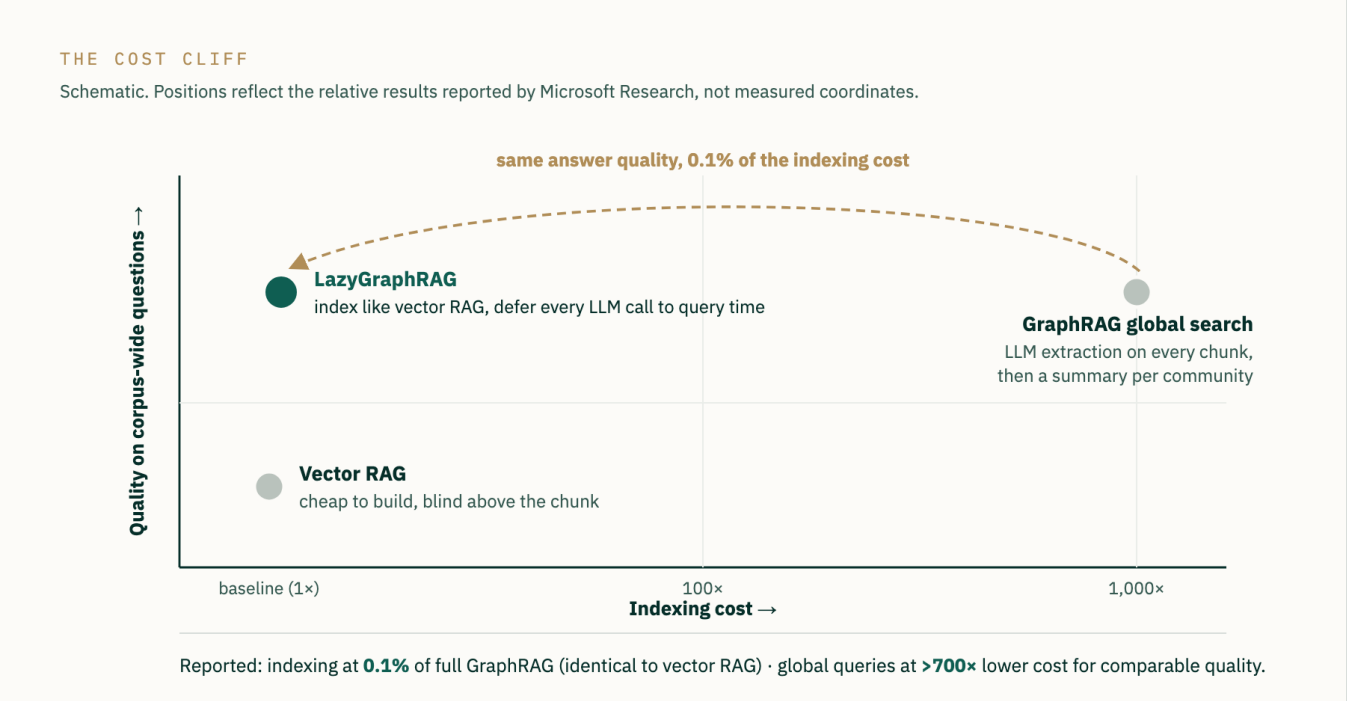

FIGURE 4 The reported shift. Deferring every language-model call to query time moves the quality of community-based retrieval down to the indexing cost of vector search.

The reported numbers:

- indexing cost identical to vector RAG, and 0.1% of full GraphRAG
- comparable answer quality to GraphRAG global search at more than 700 times lower query cost
- at 4% of global search's query cost, outperforming all competing methods on both local and global queries

Microsoft's own benchmarking suite, BenchmarkQED, puts harder numbers behind this. Against standard competitors on the same generative model, LazyGraphRAG "won all 96 comparisons, with all but one reaching statistical significance," and beat vector RAG operating with a one-million-token context window across nearly every comparison.

> Treat these as vendor-run evaluations on vendor-selected datasets. The direction is credible; the exact multiples are not something to quote to a client as your own finding.

4. The variant landscape

The architectures now in circulation differ mainly in what the graph is *made of* and what it *guides*.

- **Knowledge-graph GraphRAG** extracts a typed entity graph and retrieves by entity alignment — strongest where your domain already has an ontology

- **Community-based GraphRAG** is the Microsoft design above, with the Leiden hierarchy and local/global modes; the only architecture in common use that genuinely answers corpus-wide questions
- **Text-centric graph-guided RAG**, of which HippoRAG 2 is the reference implementation, keeps chunks as the retrieval unit and uses the graph only to guide which chunks to pull — less relational expressiveness, more precision on detail
- **Hierarchical summary structures** such as RAPTOR build a summary tree without an explicit entity graph — cheaper, and a reasonable middle position when your questions are thematic but your entities are messy
- **LazyGraphRAG** sits apart, since its graph is statistical rather than semantic and exists only to organise the search

5. What the independent evidence says

A systematic comparison across four GraphRAG families and eight benchmarks — Natural Questions, HotpotQA, MultiHop-RAG and NovelQA for question answering, plus four summarisation sets — found the win distribution splits cleanly.

Plain RAG wins on single-hop factual queries, on detail-oriented questions where fine-grained retrieval matters, and on summarisation judged against human-written references. Graph RAG wins on multi-hop reasoning, on comparison and temporal queries needing corpus-level context, and on broad summaries where coverage is the goal.

The study also names a specific weakness: community-based global search "sacrifices query-specific details," which hurts precisely the detail-centric tasks where users notice. Its recommendation is a router — send fact-based queries to vector retrieval, reasoning queries to the graph — or run both in parallel and accept the compute. The authors' framing is that the two "should not [be treated] as mutually exclusive choices."

That conclusion should shape your architecture. A Graph RAG deployment that replaces vector search is usually a downgrade for the majority of traffic. One that sits alongside it behind a classifier is where the gains are.

6. Where it breaks in production

One failure mode dominates, and it is not the retrieval algorithm.

Entity identity. The graph is only as good as its assumption that two mentions refer to the same thing. The same customer appears as *Acme Corp*, *ACME Corporation*, and a numeric account identifier across three systems; extraction dutifully creates three nodes, traversal splits the evidence between them, and the model answers confidently from a third of the facts. Entity resolution is the prerequisite, not a later optimisation, and it is the line item most often missing from Graph RAG project plans.

Compounding extraction error. Two model passes in series multiply their error rates. An entity extractor and a relation extractor that are each individually acceptable produce a graph meaningfully worse than either, and the errors are silent — a wrong edge does not throw an exception, it produces a fluent wrong answer.

Staleness. The index encodes a snapshot. Full re-indexing is expensive enough that teams defer it, and the graph drifts away from the source systems it claims to describe. Incremental update is the deciding operational feature for any variant you evaluate.

Evaluation. Comprehensiveness and diversity, the metrics the original paper optimises, are model-judged rather than ground-truth-scored. They are reasonable proxies for corpus-wide summarisation quality and they are not accuracy. Build a domain-specific evaluation set before you build the graph.

7. When to reach for it

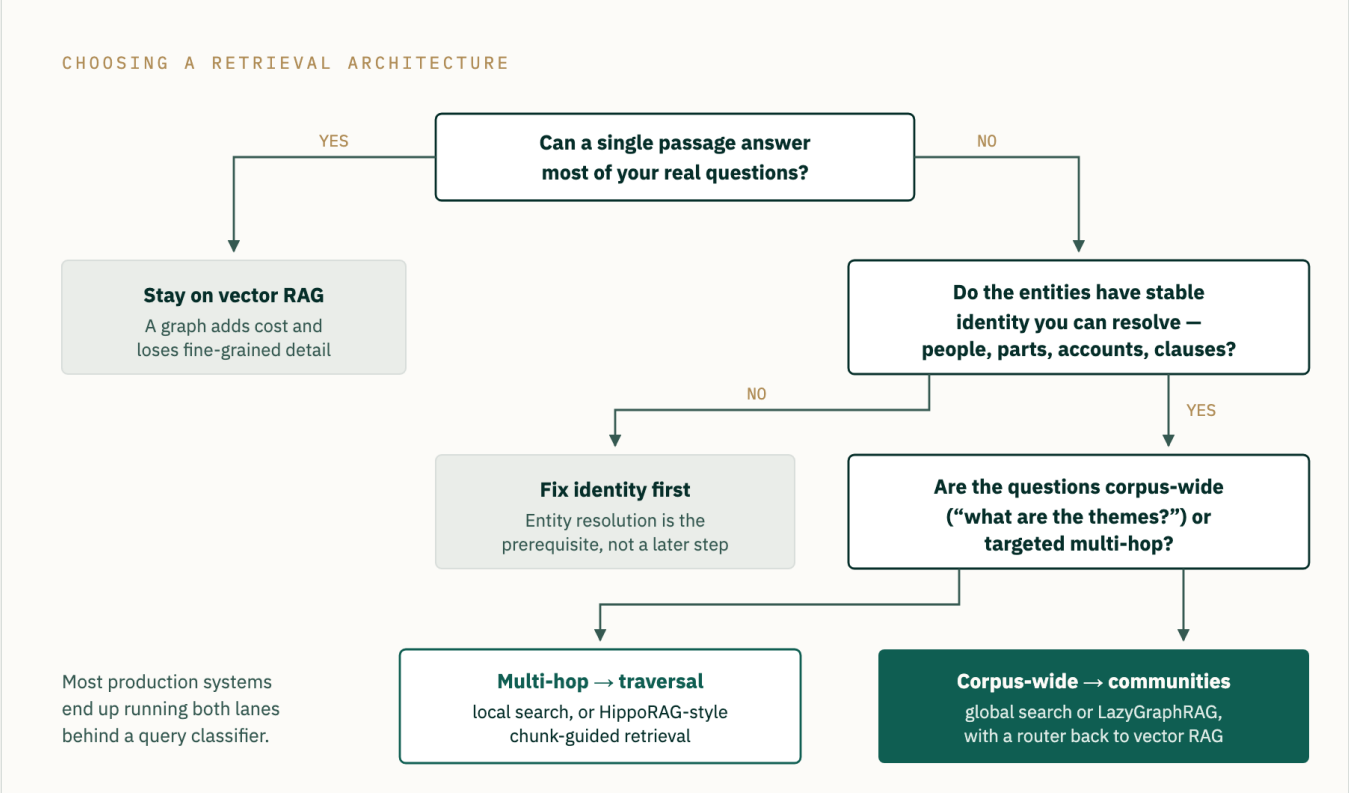

FIGURE 5 The decision in practice. Identity resolution gates everything downstream, and most production systems end up running both retrieval lanes.

Reach for a graph when the answers your users need are distributed rather than located: multi-hop questions across linked records, thematic questions over a whole corpus, comparison and timeline questions no single document contains. Reach for it when your domain has entities with stable identity — parts, accounts, people, contract clauses, controls — because that identity is what the graph is built on.

Stay with vector retrieval when most questions are answerable from one passage, when the corpus turns over faster than you can re-index, or when your entities are ambiguous enough that resolution is its own project. In that last case, resolution *is* the project, and the graph comes after.

The honest summary of the 2026 position: Graph RAG earned its reputation on a narrow but real class of question, priced itself out of reach, and has since become cheap enough that the only question left is whether your users are actually asking that kind of question. Look at the query logs before you look at the architecture diagrams.

Sources

Edge et al., *From Local to Global: A Graph RAG Approach to Query-Focused Summarization*, arXiv:2404.16130 — arxiv.org/abs/2404.16130

Peng et al., *Graph Retrieval-Augmented Generation: A Survey*, arXiv:2408.08921 — arxiv.org/abs/2408.08921

Han et al., *RAG vs. GraphRAG: A Systematic Evaluation and Key Insights*, arXiv:2502.11371 — arxiv.org/html/2502.11371v3

Microsoft Research, *LazyGraphRAG: Setting a new standard for quality and cost* — microsoft.com/en-us/research/blog/lazygraphrag-setting-a-new-standard-for-quality-and-cost

Microsoft Research, *BenchmarkQED: Automated benchmarking of RAG systems* — microsoft.com/en-us/research/blog/benchmarkqed-automated-benchmarking-of-rag-systems

Microsoft, *GraphRAG indexing architecture* — microsoft.github.io/graphrag/index/overview

Every figure in this document was drawn for it. Every quoted figure traces to one of the six sources above; where a number comes from a vendor's own evaluation, the text says so.