Arango GraphRAG Overview

What Is GraphRAG

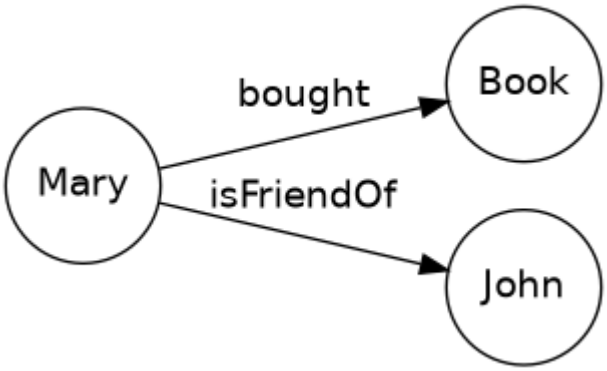
GraphRAG (Graph-based Retrieval-Augmented Generation) is Arango's approach to combining **Knowledge Graphs** with **Large Language Models (LLMs)** to deliver AI experiences that go beyond traditional text-only **RAG** systems.

It transforms document collections into structured, interconnected graphs that capture not only facts but also the relationships between concepts dramatically improving relevance, accuracy, and reasoning.

What Is a Knowledge Graph

Here is a simple Knowledge Graph which represents two pieces of information:

1. *Mary bought a book.*
2. *Mary is friend of John.*



An example of a Knowledge Graph

It has three nodes for concepts (*Mary*, *John*, *a book*) and two edges for relationships between concepts (*bought* and *is friend to*).

How GraphRAG Differs from Traditional RAG

Traditional **RAG** systems typically rely on vector similarity search over document chunks. While effective for simple lookup, they treat documents as isolated pieces of text.

GraphRAG differs in key ways:

- **Traditional RAG** retrieves semantically similar chunks and relies on the LLM to infer relationships.
- **GraphRAG** retrieves a **subgraph of explicitly connected entities and relationships**, giving

the LLM structured context to reason over.

This results in:

- Better multi-hop reasoning
- Reduced hallucinations
- Clearer traceability of answers

When and Why to Use GraphRAG

Organizations should consider **GraphRAG** when:

- Their data is **complex and interconnected** (e.g., legal, research, enterprise knowledge bases)
- Their data is stored in **unstructured** formats (PDF, Docx, other text formats)
- Queries require **contextual** or **multi-step reasoning** across documents
- **Accuracy**, **explainability**, and **trust** are critical
- **Natural-language access** to large internal knowledge collections is required

Simpler traditional **RAG** solutions may be sufficient only for basic FAQ-style or keyword-driven use cases.

End-to-End Workflow

1. Document Ingestion

- Documents are split into chunks and processed by the Importer Service
- LLMs identify entities, concepts, and relationships between them

2. Knowledge Graph Construction

- Entities become **nodes** with embeddings
- Relationships become **edges**, these include: `entity-entity`, `entity-chunk`, `chunk-document`
- The result is a domain-specific knowledge graph stored in **ArangoDB**

3. Query and Retrieval

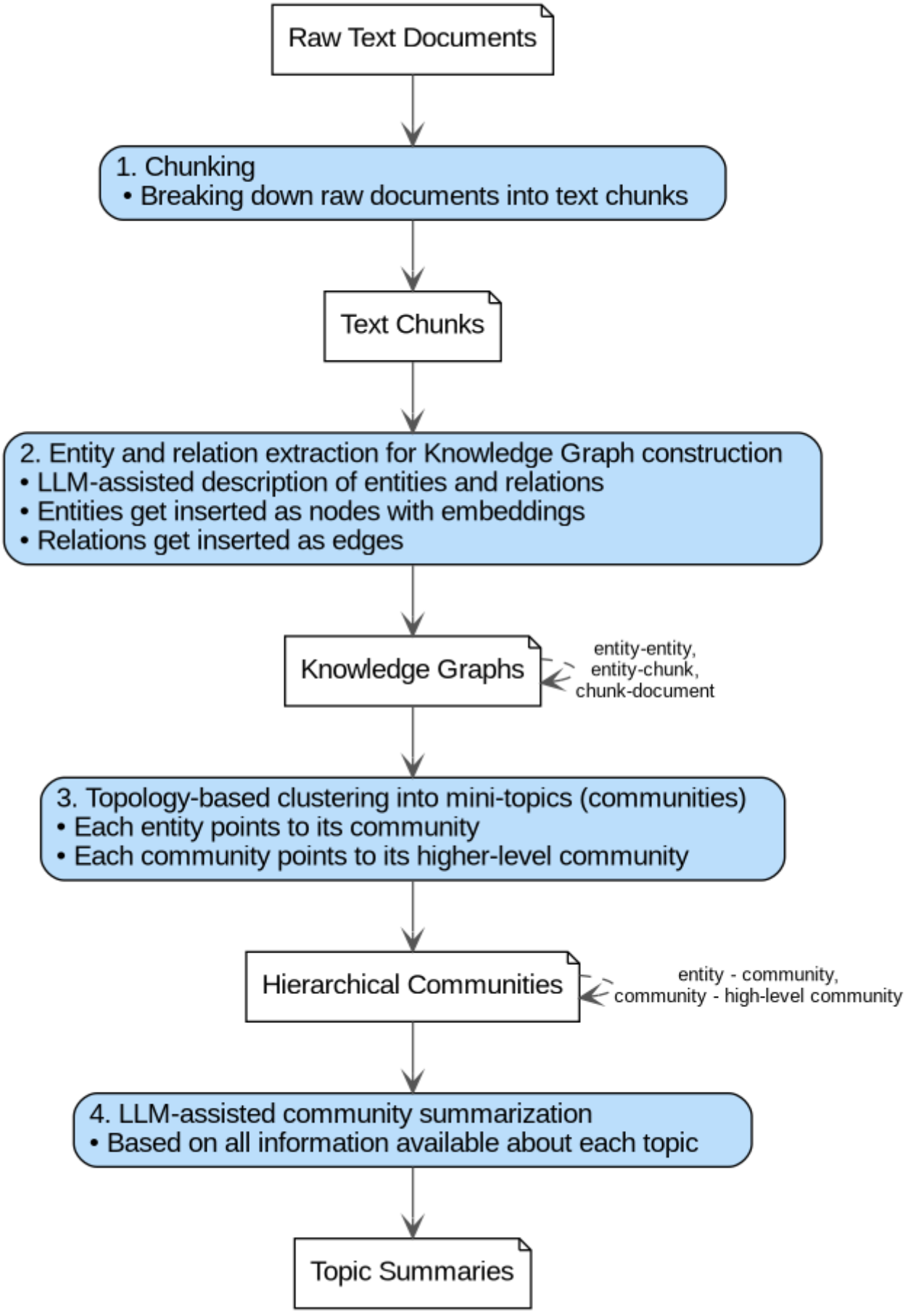
- Users ask questions in natural language
- The Retriever Service uses intelligent search and retrieval using multiple search methods to find relevant subgraphs

4. Answer Generation

- Retrieved subgraphs and context are passed to an LLM
- The LLM produces accurate, context-aware responses at the appropriate level of detail

Technical Architecture

GraphRAG is composed of modular services:



Business Use Cases

1. Enterprise Knowledge Management

Employees can query years of internal reports and documentation to receive structured, connected answers instead of isolated text snippets.

2. Research and Development

Researchers can explore relationships across studies, experiments, and findings—enabling deeper insights and faster discovery.

3. Legal and Compliance Analysis

Legal teams can analyze cases, precedents, and regulations by following explicit relationships across documents.

Data Privacy

The GraphRAG services can utilize public and private LLMs, depending on your infrastructure requirements and data governance needs.

- **Private (self-hosted) LLMs** are ideal for sensitive or regulated environments
- **Public LLMs** reduce operational overhead but require careful data handling

Deployment choices should align with compliance, security, and governance requirements.

TIP

For self-hosted option we offer **Triton Inference Server** — the backbone for running LLM and embedding models on your own machines.

For setup instructions, see Triton Inference Server and MLflow documentation.

Best Practices for Implementation and Using GraphRAG

1. Clean and curate documents before ingestion (if possible)
2. Choose LLM deployment based on privacy needs
3. Incrementally enrich the graph as new data is added
4. Continuously evaluate output quality and relevance